НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Веб-сервіс для замовлення кулінарних виробів співробітниками компаній»

Виконав:
студент IV курсу, групи КП-62
Дзенік Данило Миколайович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.,
Рибачок Наталія Антонівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
к.т.н., с.н.с. МННЦ ІТС НАНУ та МОНУ,
Євгенія Анатоліївна Савченко _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Дзеніку Данилу Миколайовичу

1. Тема проєкту «Веб-сервіс для замовлення кулінарних виробів співробітниками компаній», керівник проекту Рибачок Наталія Антонівна, ст. викладач, к.т.н., затверджені наказом по університету від «25» травня 2020 р. №1181-с

2. Термін подання студентом проєкту «16» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - збір та опис вимог до веб-сервісу для замовлення кулінарних виробів співробітниками компаній;
   - аналіз і вибір мов програмування та технологій розроблення веб-сервісів;
   - розроблення веб-сервісу для замовлення кулінарних виробів співробітниками компаній;
   - аналіз розробленого веб-сервісу для замовлення кулінарних виробів співробітниками компаній.

5. Перелік обов’язкового графічного матеріалу:
- компоненти системи (креслення);
- схема бази даних (креслення);
- інтерфейс користувача (плакат);
- архітектура системи (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	10.11.2019	
2.	Розроблення та узгодження технічного завдання	27.11.2019	
3.	Розроблення структури веб-сервісу	14.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	27.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	02.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2020	
7.	Програмна реалізація веб-сервісу	12.03.2020	
8.	Тестування веб-сервісу	20.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	04.04.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	24.04.2020	
11.	Підготовка графічної частини дипломного проєкту	12.05.2020	
12.	Оформлення документації дипломного проєкту	24.05.2020	

Студент Данило ДЗЕНІК

Керівник проєкту Наталія РИБАЧОК

АНОТАЦІЯ

Даний дипломний проект присвячено створенню веб-сервісу для замовлення кулінарних виробів співробітниками компанії.

У роботі проведено аналіз доступних програмних рішень, який показав, що існує тільки один аналог розробленому веб-застосунку. Було визначено потреби користувачів та сформульовано функціональні та нефункціональні вимоги до розробленого веб-сервісу. Було проведено аналіз технологій розроблення веб-додатків та обрано найбільш ефективні з них для використання в роботі.

Веб-сервіс містить динамічно-оновлювані списки кулінарних виробів, які обираються користувачами-співробітниками для замовлення. На основі цих замовлень постачальники можуть аналізувати, які страви мають більше попиту серед користувачів. Адміністратору веб-сервісу надано можливість аналізувати постачальників на основі замовлень та контролювати списки співробітників-користувачів, постачальників та кулінарних виробів.

У даному дипломному проекті розроблено: архітектуру веб-сервісу та структуру БД, програмні модулі для авторизації користувачів, введення та аналізу даних, дизайн веб-сторінок.

ABSTRACT

This diploma project is dedicated to the creation of a web service for ordering culinary products by the company's employees.

In this work, a comparative analysis of existing solutions showed that there is only two analogues of the developed web application. In addition, the functional and non-functional requirements for the developed web service were analyzed. The existing technologies of web application development are considered and the most effective ones for use in product development are selected. The structure of the web service and the architecture of the database have been developed.

The developed web service contains dynamically updated lists of culinary products that are selected by users to order. Based on these orders, suppliers will be able to analyze which dishes are most in demand among users. And most importantly, the administrators of this web service were given the opportunity to analyze suppliers based on user orders and the ability to control lists of users, suppliers and culinary products.

In this diploma project has developed: user authorization, architecture of client and server part of web service, software modules for work with data analysis and design of web pages.

ДП.045440-01-90 Веб-сервіс для замовлення кулінарних виробів співробітниками компаній. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Веб-сервіс для	5	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. Технічне		
	завдання		
ДП.045440-03-81	Веб-сервіс для	51	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. Пояснювальна		
	записка		
ДП.045440-04-51	Веб-сервіс для	4	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. Програма та		
	методика тестування		
ДП.045440-05-34	Веб-сервіс для	10	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. Керівництво		
	користувача		
ДП.045440-06-99	Веб-сервіс для	1	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. Архітектура		
	системи. Компоненти		
	платформи		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Веб-сервіс для	1	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. База даних		
	системи. Схема даних		
ДП.045440-08-98	Веб-сервіс для	1	
	замовлення кулінарних		
	виробів співробітниками		
	компаній. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ ЗАМОВЛЕННЯ КУЛІНАРНИХ ВИРОБІВ СПІВРОБІТНИКАМИ КОМПАНІЙ

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Данило ДЗЕНІК

2019

ЗМІСТ

1. Найменування та галузь застосування 3

2. Підстава для розроблення 3

3. Призначення розробки 3

4. Вимоги до програмного продукту 3

5. Вимоги до проєктної документації 4

6. Етапи проєктування 4

7. Порядок тестування розробки 5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сервіс для замовлення кулінарних виробів співробітниками компаній.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розроблюване програмне забезпечення призначене автоматизувати процес замовлення кулінарних виробів співробітниками компаній для організації харчування в компаніях.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб-сервіс повинена забезпечувати такі основні функції:

- авторизований доступ до веб-сервісу;
- можливість адміністратору додавати нових користувачів та постачальників;
- можливість користувачам замовляти кулінарні вироби;
- можливість постачальникам додавати, редагувати та видаляти кулінарні вироби;
- можливість постачальникам переглядати замовлення користувачів;
- можливість постачальникам генерувати звіти замовлень;
- можливість для користувача фільтрувати кулінарні вироби за

постачальником та видом страви.

Розробку серверної частини виконати на платформі Node.js з використанням фреймворку Express, клієнтської – з використанням фреймворку React.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

- пояснювальна записка;
- програма та методика тестування;
- керівництво користувача;
- креслення:
  - «Компоненти платформи»;
  - «Структура бази даних».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою проєкту……10.11.2019

Розроблення та узгодження технічного завдання……....27.11.2019

Розроблення структури веб-сервісу…….....14.12.2019

Підготовка матеріалів першого розділу проєкту…….27.12.2019

Розроблення дизайну сторінок та графічних елементів……..02.02.2020

Підготовка матеріалів другого розділу проєкту……...19.02.2020

Програмна реалізація веб-сервісу…….....12.03.2020

Тестування веб-сервісу...……....20.03.2020

Підготовка матеріалів третього розділу проєкту…….04.04.2020

Підготовка матеріалів четвертого розділу проєкту…….24.04.2020

Підготовка матеріалів графічної частини проєкту…….....12.05.2020

Оформлення технічної документації проєкту……..…24.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-СЕРВІС ДЛЯ ЗАМОВЛЕННЯ КУЛІНАРНИХ ВИРОБІВ СПІВРОБІТНИКАМИ КОМПАНІЙ

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Данило ДЗЕНІК

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 3
ВСТУП 6
1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 8
1.1. Обґрунтування вибору критеріїв оцінювання програмних рішень 8
1.2. Аналіз існуючих програмних рішень 9
1.3. Висновки за результатами аналізу 12
2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ 17
2.1. Обґрунтування вибору типу програмного забезпечення 19
2.2. Вибір системи керування базами даних 24
2.3. Вибір мови програмування для розроблення серверної частини 24
2.4. Вибір технології для розроблення клієнтської частини 27
3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ СИСТЕМИ 28
3.1. Загальний опис програми 28
3.2. Вимоги до веб-сервісу 30
3.3. Архітектура програмного забезпечення 31
3.4. Структура та опис бази даних 37
4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-СЕРВІСУ 42
4.1. Особливості реалізації 42
4.2. Тестування веб-сервісу 44
4.3. Порівняння розробленого веб-сервісу з аналогами 45
4.4. Рекомендації щодо подальшого вдосконалення 46
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 49
ДОДАТКИ 51

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Авторизація – це процедура надання користувачу визначених повноважень у системі.

Мультипарадигмальна мова програмування – мова програмування з підтримкою кількох стилів написання та організації структури програмного забезпечення.

Веб-фреймворк – каркас, призначений для створення динамічних веб-додатків та сервісів, що спрощує розробку та зменшує дублювання коду завдяки наявності готових компонентів.

Веб-браузер – програмне забезпечення для під'єднаного до мережі Інтернет пристрою, що дає можливість взаємодії з даними на гіпертекстовій веб-сторінці.

Модальне вікно – діалогове вікно, що блокує роботу користувача з додатком, доки користувач не закриє вікно після введення або отримання певної інформації.

Веб-сервіс – ідентифікована веб-адресою програмна система зі стандартизованими інтерфейсами.

Скриптова мова – мова програмування, розроблена для запису скриптів, послідовностей операцій, які користувач може виконувати на комп'ютері.

Хмарні сервіси – новітній вид мережевих послуг, які дозволяють інформаційними засобами віртуального середовища розширити програмно-технічні ресурси комп'ютерного пристрою користувача.

Громада – група людей, об'єднаних спільністю становища, інтересів, що ставить перед собою певні спільні завдання, має спільні прагнення, цілі та структури.

Рендеринг – конвертація шаблону в готову HTML-сторінку.

БД – база даних.

IT – інформаційні технології.

ПЗ − програмне забезпечення.

ОС – операційна система.

Back end − програмно-технічна частина ПЗ.

Front end − інтерфейс для взаємодії між користувачем і ПЗ.

ООП – об’єктно-орієнтоване програмування.

ПК – персональний комп’ютер.

HR − Human Resources (менеджер з персоналу).

UI − User Interface (інтерфейс користувача).

SPA − Single Page Application (односторінковий інтерфейс).

API − Application Program Interface (прикладний програмний інтерфейс).

DDL − Data Definition Language (мова опису даних).

DML − Data Manipulation Language (мова маніпулювання даними).

DOM − Document Object Model (об'єктна модель документа).

MVC − Model-View-Controller (модель–представлення–контролер).

MVVM − Model-View-View-Model.

SEO – Search Engine Optimization (пошукова оптимізація сайту).

CSV – Comma-Separated Values (значення, розділені комою).

ACID – Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, довговічність).

CSRF – Cross Site Request Forgery (міжсайтова підробка запиту).

CSS – Cascading Style Sheets (каскадна таблиця стилів).

JS – JavaScript.

SSR – Server Side Rendering (візуалізація на стороні сервера).

HTML – HyperText Markup Language (мова розмітки гіпертексту).

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту).

JSON – JavaScript Object Notation (запис об'єктів JavaScript).

UML – Unified Modeling Language (уніфікована мова моделювання).

REST – Representational State Transfer (передача стану подання).

SASS – Syntactically Awesome Style Sheets (синтаксично дивовижні таблиці стилів).

Wi-Fi – Wireless Fidelity (бездротова правдивість відтворення).

JWT – JSON Web Token (Стандарт токена доступу на основі JSON).

ВСТУП

Приймаючи співробітників на роботу, компанії доволі часто обіцяють їм, окрім заробітної плати, ще й умови, які зазвичай називають бонусами або додатковими благами. Це робиться для того, щоб зацікавити нових та підтримувати інтерес до компанії вже працюючих співробітників. Існує багато статей і навіть книг про те, як компанії рівня Google та Facebook утримують персонал. Тобто склалася така тенденція – компанії впроваджують додаткові блага, щоб приваблювати й утримувати персонал. Надання додаткових благ дає змогу працівникам зекономити гроші, покращити своє робоче самопочуття та підтримувати кращий баланс між роботою та життям. Це призводить до високого рівня мотивації, а також допомагає підтримувати довгострокові відносини з працівниками, які відчувають, що їхня компанія піклується про них, а взамін працівники докладають усіх зусиль, щоб піклуватися про свою компанію.

На сьогоднішньому ринку найму, щедрий пакет бонусів має важливе значення для залучення та збереження найкращих талантів. Згідно з опитуванням Glassdoor [1] за 2015 рік, близько 60% людей повідомляють, що пільги та додаткові блага є головним фактором при розгляді питання про те, чи приймати пропозицію на роботу. Опитування також встановило, що 80% працівників обирають додаткові блага замість підвищенням зарплати.

Одним з основних благ, на яку звертають увагу робітники - це оплата обідів з боку компанії. Під часу обідньої перерви співробітник хоче відпочити і якісно поїсти, при цьому витрати мінімум своїх ресурсів (як часу, так і грошей). А так як цей процес відбувається кожен робочий день, то додаткове благо з боку компанії на оплату обідів значно економить власні ресурси співробітнику. Тому співробітники часто віддають перевагу компаніям з додатком благом на оплату обідів.

Програмний продукт, який розробляється в рамках бакалаврського дипломного проєкту, створюється для організації харчування в офісах.

Програмне рішення буде представлено у вигляді веб-сервісу, який надає можливості для ефективного та зручного замовлення співробітниками кулінарних виробів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Перед пошуком існуючих рішень було вирішено зібрати основні потреби компаній, які організовують харчування для своїх співробітників. Зокрема, автоматизована системи було задовольняти такі потреби:

- підтримка основних веб-браузерів (Firefox, Chrome, Safari та Microsoft Edge);
- розділення прав доступу до системи для різних типів користувачів;
- відсутність функцій реєстрації у системи та можливість тільки адміністратору додавати нових користувачів;
- можливість реєстрування замовлень користувачів;
- можливість робити замовлення не раніше наступного тижня;
- можливість додавання власних постачальників;
- можливість аналізу зареєстрованих замовлень.

1.1. Обґрунтування вибору критеріїв оцінювання програмних рішень

В процесі аналізу вимог до системи було виділено наступні функціональні вимоги:

1. Наявність авторизації та розділення відповідальності на декілька ролей користувачів: адміністратор, постачальник та користувач-співробітник.
2. Адміністратор має доступ до редагування списку постачальників.
3. Адміністратор має можливість додавання, редагування та видалення користувачів.
4. Адміністратор має змогу отримати інформацію про замовлення користувачів по кожному постачальнику.
5. Постачальник має можливість додавання, редагування та видалення кулінарних виробів.

6. Постачальник має можливість аналізу кулінарних виробів по кількості замовлень користувачів.
7. Співробітник компанії має можливість замовляти страви.
8. Співробітник компанії має можливість редагувати вже зроблені замовлення.
9. Співробітник компанії має можливість відфільтрувати список кулінарних виробів за постачальником, видом страви та за назвою.
10. Співробітник компанії має можливість додавати кулінарний виріб у список улюблених страв.

Також було визначено нефункціональні вимоги:

1. Підтримка основних браузерів (Firefox, Chrome, Safari та Microsoft Edge). Перевагою може бути наявність мобільної версії веб-додатку для користувачів iPhone та Android.
2. Наявність зручного і зрозумілого інтерфейсу додатку для ефективної роботи з додатком.
3. Генерування списку замовлень у форматі CSV.
4. Модульна архітектура системи.
5. Динамічне оновлення даних.

1.2. Аналіз існуючих програмних рішень

Проаналізувавши готові програмні рішення з проблеми організації харчування для співробітників компанії, було помічено, що кожне з цих програмних забезпечень має недоліки. Основним серед них є те, що для роботи із відповідним ПЗ необхідно витратити чимало часу, а в деяких випадках не обійтись навіть без спеціальних технічних знань. Нижче наведено найбільш поширені сервіси замовлень їжі в офіс, а також їх переваги та недоліки.

1.2.1. Meido

Meido – єдине у своєму роді програмне забезпечення для організації харчування в офісах, яке спрощує весь процес, починаючи із замовлення до доставки продукції і аналізу витрат [2].

Переваги ПЗ:

- наявність декількох мов на платформі;
- наявність інтерфейсів для кожного типу користувача;
- можливість додавання власних постачальників;
- можливість додавати нових співробітників компанії через адміністратора;
- захист персональних даних.

Недоліки ПЗ:

- відсутність SPA підходу в клієнтській частині;
- незручний інтерфейс користувача;
- відсутність динамічного оновлення даних;
- відсутність можливості аналізу замовлень;
- відсутність можливості підключення до системи за допомогою API;
- відсутність можливості для користувача сортування та фільтрації страв за категоріями;
- відсутність можливості для користувача збереження улюблених страв.

1.2.2. JSolutions

JSolutions – це хмарна система для автоматизації управлінських та облікових завдань підприємств, яке дозволяє повністю автоматизувати бізнес-процеси серед яких і організація харчування на підприємстві.

Переваги ПЗ:

- наявність інтерфейсів для кожного типу користувача;
- можливість додавання кулінарних виробів у список улюблених;
- фільтрація страв за великою кількістю критеріїв;

- співробітник компанії має можливість відразу оплатити страву;
- наявність декількох мов на платформі;
- можливість генерації звітів у багатьох форматах;
- різні види візуалізації сервісу (настільний додаток, веб-додаток, мобільний застосунок);
- SPA підхід для реалізації клієнтської частини.

Недоліки ПЗ:

- незручний інтерфейс користувача;
- відсутність динамічного оновлення даних;
- відсутність можливості аналізу замовлень;
- відсутність додавання власних постачальників.

1.3. Висновки за результатами аналізу

За результатами огляду існуючих рішень проблеми організації харчування в офісах можна відзначити, що жодне з них цілком не відповідає поставленим вимогам. В табл. 1 узагальнено відповідність проаналізованих програмних рішень критеріям оцінювання автоматизованої системи з точки зору функціональних вимог.

Таблиця 1

Порівняльна характеристика існуючих рішень

Назва додатка	1	2	3	4	5	6	7	8	9	10
Meido	+	+	+	–	+	–	+	–	–	–
JSolution	+	–	+	–	–	–	+	–	+	+

Як показує таблиця, програмний продукт Meido та JSolution лише частково відповідають поставленим вимогам.

Найбільшу кількість функціональних переваг надає програмний засіб Meido. На жаль, додаток має незручний UI та обмежену кількість функціональностей для співробітника компанії.

Програмний засіб JSolution надає більше можливостей для співробітника компанії: фільтрація за багатьма параметрами; динамічне оновлення; додавання страви до списку улюблених. Але в даному додатку відсутня можливість додавання власних постачальників, що є основною вимогою до системи організації харчування в офісах.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Одним із основних кроків при розробленні програмного забезпечення є вибір засобів реалізації. Для даного застосунку необхідно обрати тип додатку, мову програмування та фреймворк для розроблення, а також обрати СУБД та технології клієнтської частини.

2.1. Обґрунтування вибору типу програмного забезпечення

Настільний додаток – це комп’ютерна програма, яка працює локально на такому комп’ютерному пристрої, як настільний комп’ютер або ноутбук, тоді як для веб-програми потрібне підключення до Інтернету або якась мережа для належної роботи [3].

Різниця між настільним додатком так веб-додатком полягає в наступному [4]:

1. Доступність з різних місць та пристроїв – якщо використати настільний додаток для написання чи запису приміток, можна переглядати файли лише на пристрої, на якому він встановлений. Але якщо використати веб-додаток з тією ж метою, можна отримати доступ до всіх своїх файлів у будь-якому місці. Кожен заклад сьогодні пропонує Wi-Fi і всюди є інтернет-кафе, тому ваші файли завжди можуть бути з вами, замість того, щоб переносити їх вручну або за допомогою флешки. Але якщо ви не можете отримати доступ Інтернету, у вас не буде доступу до веб-додатку.
2. Автоматичне оновлення – на відміну від настільних додатків, які можуть потребувати постійного і навіть ручного оновлення, веб-додатки оновлюються автоматично.
3. Рівень системних та апаратних вимог – прив'язка до апаратного забезпечення є, водночас, як і основним мінусом та і плюсом настільних додатків. Ця апаратна залежність дає можливість

настільним додаткам швидко взаємодіяти з апаратним частиною машини на якій вони працюють. При цьому потрібно бути готовим до збільшення часу розробки для того, щоб розробити програмне забезпечення, що буде працювати на багатьох платформах.

Такі програмні забезпечення для нормального функціонування можуть мати багато апаратних вимог. Тим самим апаратна залежність обмежує рівень складності функціональностей настільного додатку. У більшості випадках для реалізації веб-додатку використовують клієнт-серверну архітектуру. Користувач використовує веб-браузер для роботи з додатком, який фактично і є клієнтом, і взаємодіє з доступними в Інтернеті ресурсами, включаючи процесорну та обчислювальну потужність. Такий підхід надає доступ до складних додатків, що надходять з централізованої інфраструктури, для машин з обмеженими апаратними можливостями. Крім того, використання існуючих веб-браузерів та їх мультимедійних можливостей, дозволяє розробникам створювати більш інтерактивні, багатофункціональні інтерфейси користувача.

Виходячи з наведених вище фактів, прийнято рішення реалізувати сервіс для замовлення кулінарних виробів співробітниками компанії у форматі веб-додатку.

2.2. Вибір системи керування базами даних

Для вибору системи керування даними розглянемо дві популярні моделі баз даних: NoSQL та SQL бази даних. Доволі часто NoSQL називають нереляційними, а SQL відповідно реляційними. Різниця між ними полягає в тому як вони зберігають інформацію, як спроектовані та які типи даних вони підтримують.

2.2.1. SQL

Реляційна база даних – це тип бази даних, який зберігає і забезпечує доступ до точок даних, які пов'язані один з одним. Реляційні бази даних

базуються на реляційній моделі, інтуїтивно зрозумілому, прямому способі подання даних у таблицях. У реляційній базі даних кожен рядок таблиці – це запис із унікальним ідентифікатором, який називається ключем. Стовпці таблиці містять атрибути даних, і кожен запис зазвичай має значення для кожного атрибута, що полегшує встановлення зв'язків між точками даних [5]. SQL – це стандартна мова для роботи з реляційними базами даних. SQL можна використовувати для вставки, пошуку, оновлення та видалення записів бази даних. SQL може робити безліч інших операцій, включаючи оптимізацію та обслуговування баз даних. Таким чином, це потужний інструмент, що забезпечує програмам, користувачам й обчислювальним системам доступ до інформації, що утримується в реляційних базах даних.

Переваги SQL [6]:

- Стандартизація – SQL відповідає міжнародними стандартам ISI і ANSI.
- Швидкість.
- Відсутність кодування – керувати системами баз даних дуже просто без необхідності писати значну кількість коду, використовуючи стандартний SQL;
- Кросплатформеність – SQL можна використовувати в програмах на настільних комп'ютерах, серверах, ноутбуках і навіть деяких мобільних телефонах.
- Інтерактивність – SQL може використовуватися для спілкування з базами даних та отримання відповідей на складні запитання за лічені секунди.
- Забезпечення різного подання даних – за допомогою SQL можна представити таку структуру даних, що той або інший користувач буде бачити різні їхні подання. Крім того, дані з різних частин БД можуть бути комбіновані й представлені у вигляді однієї простої таблиці.

Недоліки SQL:

- складний інтерфейс – оскільки SQL має складну структуру, певним користувачам стає важко отримати доступ до нього;
- частковий контроль – оскільки існують певні приховані правила та умови, програмісти, які використовують SQL, не мають повного контролю над базою даних;
- вартість – деякі версії SQL коштують дорого, тому програмісти не можуть отримати доступ до неї.

2.2.2. NoSQL

Бази даних NoSQL є протилежними до реляційних, оскільки вони працюють абсолютно різними методами. У нереляційній базі даних немає наборів таблиць; вона не використовує структуру стовпців і рядків для зберігання даних. У базах даних NoSQL існує величезна кількість моделей, розроблених для різних типів даних. Така база даних нагадує папку з файлами [7].

NoSQL було розроблено в основному для вирішення проблеми масштабованості, характерної для SQL. Як результат, він працює без схем і побудований на розподілених системах, що дозволяє легко масштабувати та розміщувати дані.

Однак ці переваги пов'язані з послабленням принципів ACID: атомарність, узгодженість, ізольованість, довговічність. Замість того, щоб підтримувати всі чотири параметри протягом кожної транзакції, NoSQL використовує принцип послідовності. Це означає, що якщо не буде нових оновлень для певного елемента даних протягом певного періоду часу, з часом всі звернення до нього повернуть останнє оновлене значення.

Хоча такий підхід значно збільшує час доступу та масштабованість, це може призвести до втрати даних, яка в основному залежить від підтримки сервера баз даних та якості коду програми.

Переваги NoSQL [8]:

- Масштабованість – може горизонтально рости, можливість додавати дані, не втрачаючи структури. Для того, щоб продуктивність залишалася на колишньому рівні, просто потрібно оновити обладнання, а не купувати більше серверів.
- Можливість зберігати та працювати з величезною кількістю даних.
- Вартість – дані можуть поширюватися на дешевих серверах, а продуктивність залишатиметься на найвищому рівні.
- Відсутність схем – для завантаження даних у базу даних NoSQL не потрібна схема, оскільки формат і модель можуть бути змінені в будь-який час.
- Використання системної пам'яті для зберігання даних кешу.

Недоліки NoSQL:

- менша підтримка громади у порівнянні з SQL;
- відсутність стандартизації – призводить до проблем при міграції бази даних;
- великий розмір документів.

Виходячи з описаних вище характеристик NoSQL та SQL баз даних можна побудувати наступну таблицю порівняння (табл. 2).

Таблиця 2

Порівняння SQL та NoSQL баз даних

SQL	NoSQL
Реляційна база даних	Розподілена або нереляційна база даних
Має чітко розроблену або попередньо визначену схему для структурованих даних	Має динамічну схему. Дані можна гнучко зберігати без попередньо визначеної структури

Продовження таблиці 2

Вертикальна масштабованість	Горизонтальна масштабованість
Не підходить для ієрархічного зберігання даних	Бази даних NoSQL найкраще підходять для ієрархічного зберігання даних, оскільки це відповідає методу ключ-значень для зберігання даних
Відповідає властивостям ACID	Не відповідає властивостям ACID
Додавання нових даних в базу даних вимагає внести деякі зміни у схему, що ускладнює процес	Нові дані можна легко вставити в бази даних, оскільки це не вимагає жодних попередніх кроків

Виходячи з ймовірної схеми даних розроблюваного дипломного проекту та наведених вище фактів прийнято рішення використовувати NoSQL для зберігання даних сервісу.

2.3. Вибір мови програмування для розроблення серверної частини

Після аналізу функціональних особливостей даного веб-сервісу було зроблено висновок про вибір клієнт-серверної архітектури, а саме: модуль клієнтської частини для реалізації користувацького інтерфейсу та модуль серверної частини, що відповідатиме за взаємодію із зовнішніми сервісами та базою даних. Для реалізації серверної частини веб-додатку було вирішено обрати інтерпретовану мову програмування з підтримкою ООП, великою кількістю стандартних бібліотек та з наявністю вибору фреймворку для побудови веб-серверу.

2.3.1. Python

Python – інтерпретована мова програмування високого рівня з динамічною типізацією. Мова програмування Python в основному передбачає імперативне та об'єктно-орієнтоване функціональне програмування. Вона містить в собі велику і комплексну стандартну

бібліотеку, яка має динамічні функції та автоматичне управління пам'яттю [9].

Основні характеристики мови Python [10]:

- широка підтримка бібліотек – Python надає велику кількість стандартних бібліотек, які включають такі сфери, як операції з рядком, Інтернет, інструменти веб-служб, інтерфейси операційних систем та протоколи. Більшість широко використовуваних функцій програмування вже реалізовані в мові Python, що дозволяє програмістам не писати власні їх реалізації;
- доступність – Python безкоштовний, тому приватні особи, невеликі компанії чи великі організації можуть використовувати вільні наявні ресурси для створення додатків. Python популярний і широко використовується, тому він має кращу підтримку у громаді;
- кросплатформність – код Python може працювати на будь-якій операційній системі;
- продуктивність – оскільки кроку компіляції немає, цикл редагування та тестування коду надзвичайно швидкий.

Порівняно з іншими мовами програмування Python є найбільш широко використовуваним. Найважливіші переваги мови Python полягають у тому, що її легко читати та вивчити. Встановлення пакетів та написання програмного забезпечення на Python менш складні, ніж у C або C ++. Деякі інші переваги програмування Python полягають у тому, що жоден комп'ютерний вірус не може спричинити помилку сегментації, оскільки в Python немає понять “посилань”. Важливою перевагою мови Python перед традиційними мовами програмування є те, що вона має широке застосування та сприйняття. Тому вона інтенсивно використовується як вченими, інженерами та математиками, для створення анімації для фільмів чи в машинному навчанні.

2.3.2. JavaScript

JavaScript (часто просто JS) – це легка, інтерпретована, об'єктно-орієнтована мультипарадигмальна мова програмування, і найбільш відома як скриптова мова для веб-сторінок, але може також використовуватись у багатьох середовищах, які не є браузером [11]. JavaScript відповідає специфікації ECMAScript, тому є динамічною і підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування. JavaScript інтерфейси прикладного рівня програмування (API) дозволяють працювати з датами, текстом, DOM, стандартними структурами даних та регулярними виразами [12, 13].

Переваги JavaScript [14]:

- швидкість – інтерпретованість мови дозволяє відмовитись від компіляції, необхідний іншими мовами програмування;
- простота – JavaScript синтаксис є відносно простим у реалізації та вивченні;
- універсальність – JavaScript чудово взаємодіє з іншими мовами програмування і може використовуватися для розробки настільних додатків, веб-серверів та веб-браузерів.

Недоліки JavaScript:

- безпека – в деяких випадках JavaScript код можна використати в зловмисних цілях, оскільки він виконується в основному на комп’ютерах користувачів. Крім того, на веб-сайті доволі легко розмістити код який може загрожувати безпеці даних;
- підтримка браузерів – хоча скрипти на стороні сервера завжди дають однаковий результат, різні браузери іноді інтерпретують код JavaScript по-різному.

У таблиці 3 наведено результати порівняння мов програмування для розроблення серверної частини.

Таблиця 3

Порівняння мов програмування для розроблення серверної частини

	Інтерпретована модель виконання	Функціональна парадигма	Велика кількість бібліотек	Фреймворк для веб-серверу
Python	+	+\–	+	+
JavaScript	+	–	+	+

Виходячи з наведених вище фактів прийнято рішення обрати для серверної частини веб-додатку мову JavaScript, яка не тільки відповідає всім встановленим вимогам, а й дозволяє повторно використати деякі частини коду з клієнтської частини системи.

2.4. Вибір технології для розроблення клієнтської частини

Основними вимогами до технології (фреймворку) для реалізації клієнтської частини є:

- компонентний підхід до архітектури;
- наявність великої кількості бібліотек або модулів;
- швидкість оновлення стану сторінки.

2.4.1. Vue.js

Vue.js – це прогресивний JavaScript фреймворк, який використовується для побудови інтерфейсів користувача. На відміну від деяких інших популярних фреймворків, його не підтримує жодна велика технологічна компанія: хоча React був створений та підтримується Facebook, Angular підтримується Google, Vue.js повністю створений та підтримується громадою. Основна увага в ньому приділяється на шар перегляду (інтерфейс користувача, сторінки та інші візуальні елементи), що дозволяє фреймворку легко інтегруватися в існуючі проекти, але це також вдалий вибір, якщо ви створюєте складну односторінкову програму (SPA) за умови, поєднуєте його із сучасними інструментами [14].

Переваги [15]:

- Масштабованість – Vue.js можна використовувати для побудови інтерактивних частин, які інтегруються за допомогою інших технології, а також для величезного модульного SPA.
- Невеликий розмір фреймворку.
- Двостороннє прив'язування даних – зв'язок між переглядом інтерфейсу користувача (UI) та оновленнями даних моделі. Зв'язані компоненти містять дані, які можна час від часу оновлювати. За допомогою двосторонньої прив'язки даних простіше оновлювати пов’язані компоненти та відстежувати оновлення даних. Дана привязка даних успадкована від Angular.
- Однофайлові компоненти – кожен фрагмент веб-сторінки є компонентом. Компоненти представляють ізольовані елементи вашого інтерфейсу. У Vue.js компоненти можна записувати JavaScript, HTML та CSS, не поділяючи їх на окремі файли.
- Екосистема – має власні інструменти для налагодження браузера, SSR та менеджер станів.
- Складність – Vue.js доволі легко вивчити завдяки простому синтаксису.
- Коротка та проста документація.
- Велика підтримка громади.

Недоліки:

- Мовний бар’єр – завдяки популярності Vue.js у Китаї значна частина її змісту та обговорень ведеться китайською.
- Стабільність – розмір спільноти та розроблювальної команди Vue.js все ще незрівнянний із більш зрілим Angular. Також у фреймворка відсутня фінансова підтримка великих підприємств.
- Обмежені ресурси – хоча екосистема досить широка і є всі необхідні інструменти, щоб почати розробку з Vue.js, вона все ще

не така велика, як React або Angular. Якщо точніше, кількість плагінів, доступних для React, у сотні разів більша, а ніж у Vue.js. Але це може бути лише питанням часу.

2.4.2. Angular

Angular – це веб-система з відкритим вихідним кодом, заснована на JavaScript, в основному підтримується Google. Angular забезпечує структуру для архітектури MVVM та MVC на стороні клієнта, яка спрямована на спрощення як розробки, так і тестування таких додатків. Поєднуючи введення залежностей, декларативні шаблони та найкращі інтегровані практики, він вирішує майже всі проблеми при створенні веб-додатку [16].

Переваги[17]:

- MVC архітектура – ізолювання логіки програми від рівня інтерфейсу. Контролер отримує всі запити на веб-додаток і працює з моделлю, щоб підготувати будь-які дані, необхідні для перегляду. Перегляд використовує дані, підготовлені контролером, і відображає остаточний результат;
- Двостороння прив'язка даних – Angular був побудований з архітектурою MVC тому, коли дані в моделі змінюються, відображення також змінюється. Двостороння прив'язка даних дозволяє інженерам скоротити час розробки, оскільки не потрібно писати додатковий код для постійної синхронізації відображення та моделі.
- Впровадження залежностей – робить Angular компоненти більш багаторазовими, легшими в управлінні та тестуванні.
- Велика підтримка громади.
- Використання директив – дозволяє зберігати скрипти та сторінки HTML надзвичайно організованими. Директиви дозволяють створювати незалежні компоненти, стискаючи певні функції та використовуючи їх неодноразово. Крім набору

заздалегідь визначених директив, структура також дозволяє розробникам створювати власні директиви.

- Екосистема – присутня велика кількість пакетів, плагінів, додатків та інструментів розробки.

Недоліки:

- обмежені можливості SEO;
- складність вивчення;
- документація – відсутність повної інформації про функціональність Angular.

2.4.3. React

React – JavaScript бібліотека, що призначена для побудови користувацьких інтерфейсів. React поєднує в собі новий спосіб візуалізації веб-сторінок і швидкість JavaScript, роблячи веб-сторінки дуже динамічними та чутливими до введення даних користувачами [18]. Продукт значно змінив підхід Facebook до розробки. Після випуску бібліотеки в якості інструменту JavaScript з відкритим кодом у 2013 році вона стала надзвичайно популярною завдяки революційному підходу до програмування користувацьких інтерфейсів. React можна використовувати як базу при розробці односторінкових або мобільних додатків. Однак React стосується лише надання даних у DOM, тому створення React додатків зазвичай вимагає використання додаткових бібліотек для управління станом додатка та його маршрутизації. Redux та React Router є відповідними прикладами таких бібліотек [19].

Переваги [20]:

- процес оновлень оптимізований та прискорений – можливість використання вбудованих методів для мінімізації кількості операцій DOM, що допомагає оптимізувати процес оновлення та прискорити його;
- віртуальний DOM – прискорює швидкість оновлень. Мінімалістичні зміни, застосовані користувачем, не впливають

на інші частини інтерфейсу користувача;

- читабельність – JSX робить код компонентів читабельним. Він показує, як компоненти підключаються або поєднуються;
- прив'язка даних React встановлює умови для створення динамічних додатків;
- можливість тестування – існує велика кількість інструментів для тестування React компонентів;
- SEO – представляє досвід першого завантаження за допомогою надання серверного рендерингу та підключення обробників подій на стороні користувача;
- екосистема – присутня велика кількість пакетів, плагінів, додатків та інструментів розробки;
- велика підтримка громади – команда Facebook підтримує бібліотеку;
- компонентна ізольованість – можливість повторно використовувати компонент в різних частинах веб-додатку економить значну кількість часу розробника;
- односторонній потік даних – гарантує, що зміни дочірніх структур не впливають на їх батьків.

Недоліки:

- великий розмір бібліотеки;
- складність вивчення для початківців;
- проблем з індексуванням пошукових систем через використання ізоморфного підходу до побудови програми.

Проведений аналіз свідчить про те, що технології Angular и Vue.js лише частково відповідають встановленим вимогам на відміну від React. Виходячи з наведених вище фактів прийнято рішення обрати бібліотеку React для розроблення клієнтської частини.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ СИСТЕМИ

Для логічної та правильної роботи веб-застосунку перед початком самого розроблення необхідно правильно спроектувати та організувати структуру програмного забезпечення. Структурно-алгоритмічна організація впливає на швидкість розроблення, можливість майбутнього розширення системи, а також на швидкість роботи додатку.

3.1. Загальний опис програми

Веб-сервіс для замовлення кулінарних виробів співробітниками компанії – це сервіс, який складається з клієнтської та серверної частин. Основна задача даного програмного забезпечення полягає в тому, щоб надати змогу компаніям організувати харчування для їх співробітників. Як було вже зазначено у попередніх розділах, даний застосунок орієнтований на середній бізнес та великий бізнес, який вже має достатньо ресурсів, щоб інвестувати гроші для покращення умов харчування співробітників.

Даний додаток підтримує три типи користувачів:

- постачальник – авторизований користувач даного веб-застосунку, якому доступна лише його особиста сторінка постачальника, в якій він може переглядати та редагувати список своїх кулінарних виробів та список замовлень його страв;
- співробітник компанії – авторизований користувач даного додатку, який має доступ до головної сторінки кулінарних виробів, де він може обрати страву у постачальника, а також має доступ до особистого кабінету в якому може переглянути список вже зроблених замовлень та відредагувати їх;
- адміністратор – авторизований користувач веб-застосунку, який має окрему адмін-панель та можливість управління всім контентом на сайті: створення, видалення, редагування користувачів та постачальників; переглядати та видаляти будь-

які кулінарні вироби, а також робити будь-які зміни у замовленнях користувачів-співробітників;

- спостерігач – неавторизований користувач, який має доступ тільки до сторінки авторизування, через яку він може увійти у систему або зробити запит до адміністратора на додавання його у систему.

Функціональні можливості співробітника компанії щодо роботи із застосунком подано у вигляді діаграми на рис. 1. Нижче наведено опис основного сценарію взаємодії співробітника із додатком. Перше, куди потрапляє даний користувач – це сторінка для авторизування, на якій розміщена коротка інформація про сам продукт та його можливості. Для повноцінної роботи з даним застосунком потенційному користувачу потрібно авторизуватися. Далі він переходить на головну сторінку, де він має змогу обрати страву на даний день. Для швидкого вибору страви у співробітника компанії є можливість фільтрувати кулінарні вироби за постачальником, популярністю, рейтингом, назвою, спеціальними тегами (вегетаріанське, м’ясне, суп тощо). Слід зазначити, якщо користувачеві сподобалась дана страва, то для швидкого доступу він може додати її до списку улюблених страв. В головному меню вибору страви співробітник компанії отримує коротку інформацію про кулінарний виріб, а для більш детальної інформації йому необхідно перейти на основну сторінку страви. Після того, як співробітник компанії обрав кулінарний виріб, він отримує модальне вікно для підтвердження, де також потрібно обрати дату його отримання. Після його підтвердження користувач повертається у головне вікно вибору страв. Також він може перейти у власний кабінет та переглянути або відредагувати власні замовлення.

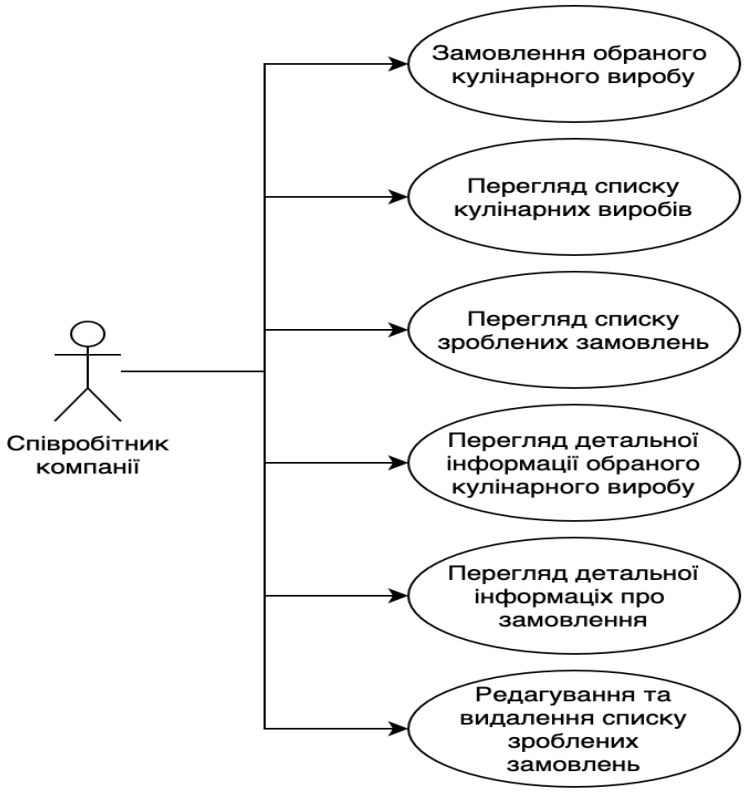


Рис. 1. Варіанти можливостей співробітника компанії

Головний акцент у даному веб-застосунку зроблений на інтуїтивно зрозумілий інтерфейс та зручну роботу для будь-якого користувача, тому програма містить мінімально необхідний набір основних функцій, які необхідні для реалізації вибору замовлення.

Розглянемо функціональні можливості адміністратора:

- керування списком користувачів (створення, редагування даних, видалення);
- керування списком постачальників (створення, редагування даних, видалення);
- керування кулінарними виробами (створення, редагування, видалення);
- керування замовленнями користувачів (видалення та редагування).

Розглянемо функціональні можливості постачальників:

- керування списком страв (створення, редагування, видалення);

– перегляд сформованого списку замовлень;

– генерація звітів замовлень.

Спостерігач – є найпростішим типом користувача у системі. Йому недоступні будь-які наміри пов'язані зі управлінням або створенням кулінарних виробів. У даного типу користувача є тільки можливість зробити запит адміністратору для реєстрації у систему.

3.2. Вимоги до веб-сервісу

Веб-сервіс для замовлення кулінарних виробів співробітниками компанії можна уявно поділити на декілька частин: особистий кабінет користувача, сторінка адміністратора, сторінка постачальника, головна сторінка сервісу, сторінка кулінарного вироба. Для кращого формулювання вимог будемо розглядати кожну частину окремо.

3.2.1. Вимоги до сторінки особистого кабінету співробітника компанії

Завданням особистого кабінету співробітника компанії є саме перегляд та редагування вже зроблених ним замовлень.

Функціональні та нефункціональні вимоги до кабінету користувача представимо у вигляді реєстру у табл. 4.

Таблиця 4

Реєстр вимог до особистого кабінету співробітника компанії

Код вимоги	Зміст вимоги	Пріоритет вимоги	Складність вимоги	Тип вимоги
С1	Можливість перегляду списку своїх замовлень в рамках певного періоду часу	Високий	Висока	Інтерфейс
С2	Можливість зміни страви певного замовлення	Середній	Висока	Функціонал
С3	Можливість видалення майбутніх замовлень	Високий	Середня	Функціонал

Продовження таблиці 4

С4	Зміна рамок періоду часу замовлень користувача	Середній	Висока	Функціонал
С5	Можливість перейти на головну сторінку користувача	Високий	Висока	Функціонал
С6	Можливість перейти на головну сторінку користувача	Високий	Висока	Інтерфейс
С7	Можливість пагінації для списку замовлень користувача	Високий	Висока	Інтерфейс

3.2.2. Вимоги до головної сторінки співробітника компанії

Основним завданням головної сторінки співробітника компанії полягає у наданні можливості легко та просто замовити кулінарний виріб.

Функціональні та нефункціональні вимоги до головної сторінки співробітника компанії представимо у вигляді реєстру у табл. 5.

Таблиця 5

Реєстр вимог до головної сторінки співробітника компанії

Код вимоги	Зміст вимоги	Пріоритет вимоги	Складність вимоги	Тип вимоги
С8	Можливість перегляду списку всіх кулінарних виробів	Високий	Висока	Інтерфейс
С9	Можливість пошуку кулінарного виробу за назвою	Високий	Середня	Функціонал
С10	Можливість змінювати режим відображення списку страв (галерея або список)	Низький	Середня	Інтерфейс

Продовження таблиці 5

С11	Можливість фільтрації за складом страви	Низький	Висока	Функціонал
С12	Можливість фільтрації списку страв за спеціальними тегами (вегетаріанська або м’ясна страва)	Високий	Висока	Функціонал
С13	Можливість фільтрації списку страв за постачальниками	Високий	Низька	Функціонал
С14	Можливість додавання страви до власного списку улюблених страв	Середній	Висока	Функціонал
С15	Можливість зробити замовлення страви на певну дату	Високий	Висока	Функціонал
С16	Можливість перейти на окрему сторінку кулінарного виробу для більш детальної інформації	Низький	Висока	Інтерфейс
С17	Можливість перейти в особистий кабінет користувача	Високий	Висока	Інтерфейс
С18	Можливість фільтрації списку страв за популярністю	Низький	Середня	Функціонал
С19	Можливість відображення тільки списку улюблених страв	Середній	Висока	Інтерфейс
С20	Доступність сторінки тільки для авторизованих користувачів	Низький	Середня	Функціонал

Продовження табл. 5

С21	Можливість пагінації для списку кулінарних виробів	Середній	Висока	Інтерфейс

3.2.3. Вимоги до сторінки адміністратора

Основну частину функціональну наповнення закладено також в панелі адміністратора. Функції налаштування, додавання, та видалення співробітників компанії та постачальників, управління замовленнями – всі ці обов'язки закладені у роль адміністратора веб-застосунку. Звідси витікають і основні вимоги до створення та зручного наповнення адмін-панелі для керування ПЗ. Всі ці вимоги винесені до реєстру вимог, що знаходиться у табл. 6.

Таблиця 6

Реєстр вимог до сторінки адміністратора

Код вимоги	Зміст вимоги	Пріоритет вимоги	Складність вимоги	Тип вимоги
А1	Доступність сторінки тільки для адміністратора	Високий	Висока	Функціонал
А2	Відображення сторінки відповідно до дизайну	Середній	Середня	Інтерфейс
А3	Можливість підтвердження запитів на реєстрацію користувачів	Низький	Висока	Функціонал
А4	Можливість додавання, редагування та видалення користувачів	Високий	Висока	Функціонал
А5	Можливість управління замовленнями користувачів (редагування, видалення)	Середній	Висока	Функціонал

Продовження табл. 6

А6	Можливість додавання, редагування та видалення постачальників	Високий	Висока	Функціонал
А7	Можливість редагування та видалення кулінарних виробів постачальника	Середній	Середня	Функціонал
А8	Можливість пагінації списку користувачів та списку постачальників	Середній	Низька	Інтерфейс

3.2.4. Вимоги до сторінки постачальника

Основним завданням сторінки постачальника полягає у наданні інтерфейсу для зручного управління даними кулінарних виробів, а також для відображення замовлень користувачів.

Функціональні та нефункціональні вимоги до сторінки постачальників представимо у вигляді реєстру у табл. 7.

Таблиця 7

Реєстр вимог до сторінки постачальника

Код вимоги	Зміст вимоги	Пріоритет вимоги	Складність вимоги	Тип вимоги
П1	Доступність сторінки тільки для постачальника.	Високий	Висока	Функціонал
П2	Відображення сторінки відповідно до дизайну	Середній	Середня	Інтерфейс
П3	Можливість додавання, видалення, редагування кулінарних виробів	Високий	Висока	Функціонал
П4	Можливість перегляду списку замовлень користувачів у даного постачальника	Високий	Висока	Інтерфейс

Продовження табл. 7

П5	Можливість видалення та редагування замовлення користувача	Високий	Висока	Функціонал
П6	Можливість генерації звіту замовлень у форматі CSV	Високий	Висока	Функціонал
П7	Можливість сортування списку замовлень за кулінарним виробом, датою, користувачем	Середній	Низька	Інтерфейс
П8	Можливість отримання списку замовлень в певних часових межах	Середній	Середня	Інтерфейс
П9	Можливість пагінації для списку замовлень та списку страв	Середній	Низька	Інтерфейс

У результаті проведеного аналізу було встановлено вимоги до веб-сервісу для замовлення кулінарних виробів співробітниками компанії. Всі нефункціональні та функціональні вимоги було занесено до реєстру вимог з відповідними помітками про тип. Також було визначено типи та можливості користувачів, що також представлено у реєстрі вимог. Крім цього, кожна вимога була проаналізована на важливість та складність виконання, що зазначено у реєстрі у відповідних колонках.

3.3. Архітектура програмного забезпечення

Розроблена система для замовлення кулінарних виробів співробітниками компанії реалізована у вигляді веб-сервісу, що складається з 3 основних частин: веб-серверу, клієнтської частини та бази даних.

Загальна структура розробленого сервісу зображена на рис. 2.

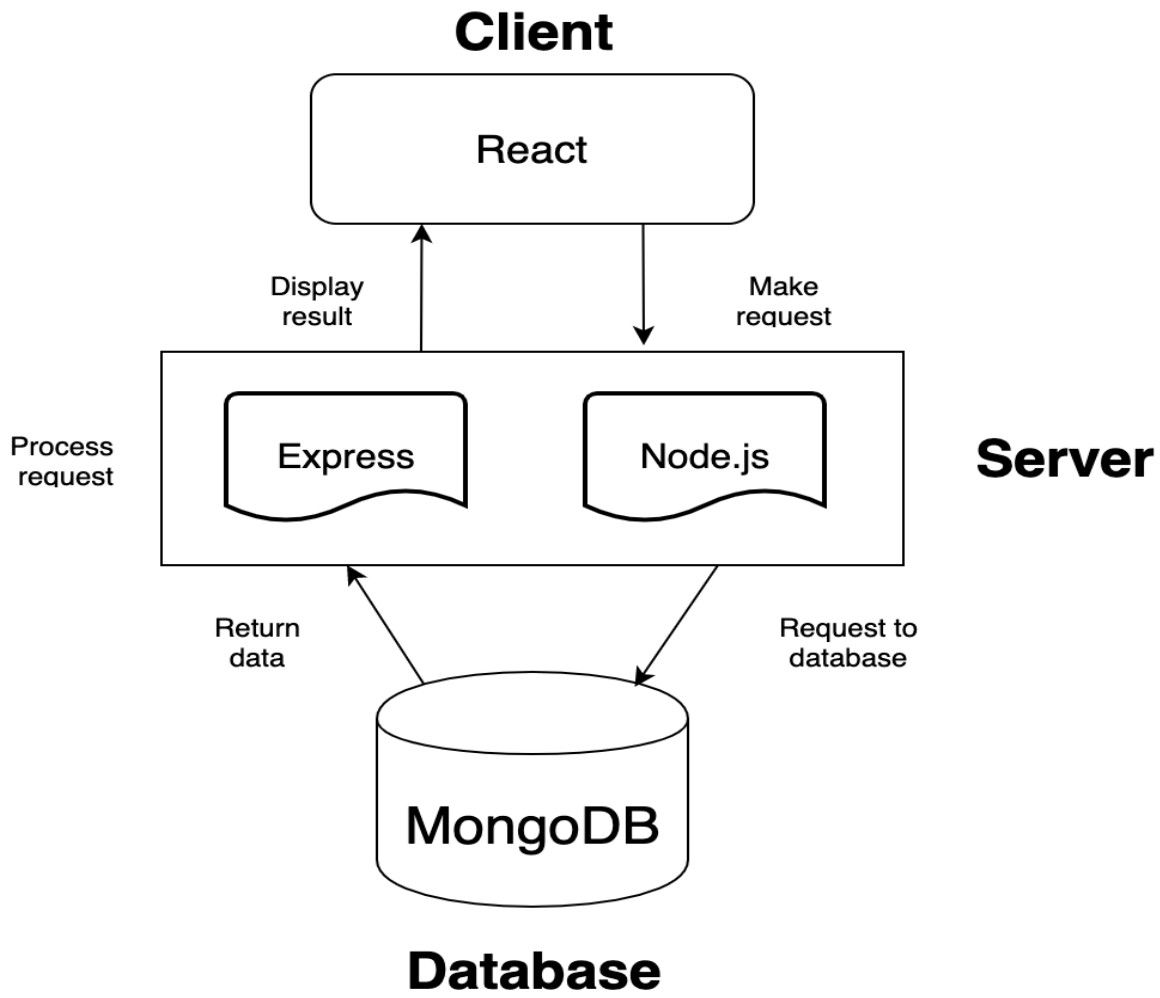


Рис. 2. Структурна схема сервісу

Веб-сервер отримує запити від клієнта через REST API, обробляє їх та повертає дані. Він реалізований в даному сервісі за допомогою фреймворку Express на платформі Node.js. Веб-сервер містить такі засоби:

- взаємодії з базою даних;
- обробки HTTP запитів;
- валідації даних;
- CRUD операції з основними сутностями системи;
- авторизації користувачів.

REST API – структура запитів для отримання коректної відповіді від серверу, яких повинен дотримуватись клієнт.

За допомогою бібліотеки об’єктного моделювання даних Mongoose відбувається взаємодія з базою даних MongoDB на серверній частині

веб-застосунку. Вона використовується для представлення об’єктів у MongoDB, а також забезпечує перевірку схем. Завдяки саме Mongoose в окремих JavaScript-файлах було створено схеми сутностей та визначено зв’язки між ними.

Веб-клієнт відповідає за відображення користувачеві сторінок веб-сервісу, які отримує з сервера. Компонентна й гнучка структуру веб-застосунку забезпечена завдяки фреймворку React, за що відповідає пакет components.

Кожен компонент є незалежним один від одного і складається з таких частин:

- JavaScript-скрипта, який описує логіку роботи компонента;
- HTML-шаблону, який відповідає за подання контенту;
- модульного SASS–стилів.

Веб-сторінка може як складатися з одного React-компонента так і поєднувати в собі декілька. Така структура допомагає організовувати гнучку логіку веб-застосунку та повторно використовувати готові елементи декілька разів.

3.4. Структура та опис бази даних

MongoDB використовується в якості нереляційної бази даних. Вона зберігає дані постачальника, кулінарні вироби, які він створює, та інформацію про користувачів та їх замовлення. Дана частина веб-застосунку відповідає за такі задачі:

- фільтрація даних за необхідними критеріями;
- пошук даних за необхідними параметрами;
- безпека даних, що зберігаються у базі.

В результаті проектування додатку було виявлено такі сутності: Customer, Supplier, Order, Dish.

Таблиця Customer зберігає інформацію про користувача та адміністратора:

- id – унікальний ідентифікатор;
- тип користувача;
- ім’я співробітника;
- прізвище співробітника;
- електронна пошта – використовується для входу в систему;
- пароль – зберігається в зашифрованому вигляді;
- улюблені кулінарні вироби;
- дата реєстрації;
- номер телефона.

Таблиця Supplier зберігає інформацію про постачальника:

- id – унікальний ідентифікатор;
- назва компанії;
- ім’я постачальника;
- прізвище постачальника;
- електронна пошта – використовується для входу в систему;
- пароль – зберігається в зашифрованому вигляді;
- дата реєстрації;
- номер телефона;
- адреса постачальника.

Таблиця Order зберігає інформацію про замовлення:

- id – унікальний ідентифікатор;
- id замовника;
- id кулінарного виробу;
- дата створення замовлення;
- дата, коли було створено замовлення.

Таблиця Dish зберігає інформацію про кулінарний виріб:

- id – унікальний ідентифікатор;
- id – постачальника;

- назва страви;
- картинка;
- опис;
- ціна;
- кількість лайків;
- кількість калорій;
- вага;
- набір тегів – (вегетаріанська, м’ясна, суп, тощо);
- дата створення.

У табл. 8 визначено відношення між сутностями системи.

Таблиця 8

Відношення між сутностями системи

Сутність 1	Сутність 2	Тип відношення
Supplier	Dish	One to many optional
Customer	Dish	Many optional to many optional
Customer	Order	One to many optional
Dish	Order	One to many optional

Описаним сутностям відповідають таблиці Supplier, Customer, Dish, Order. На рис. 3 зображено ERD-діаграму бази даних розробленого додатку.

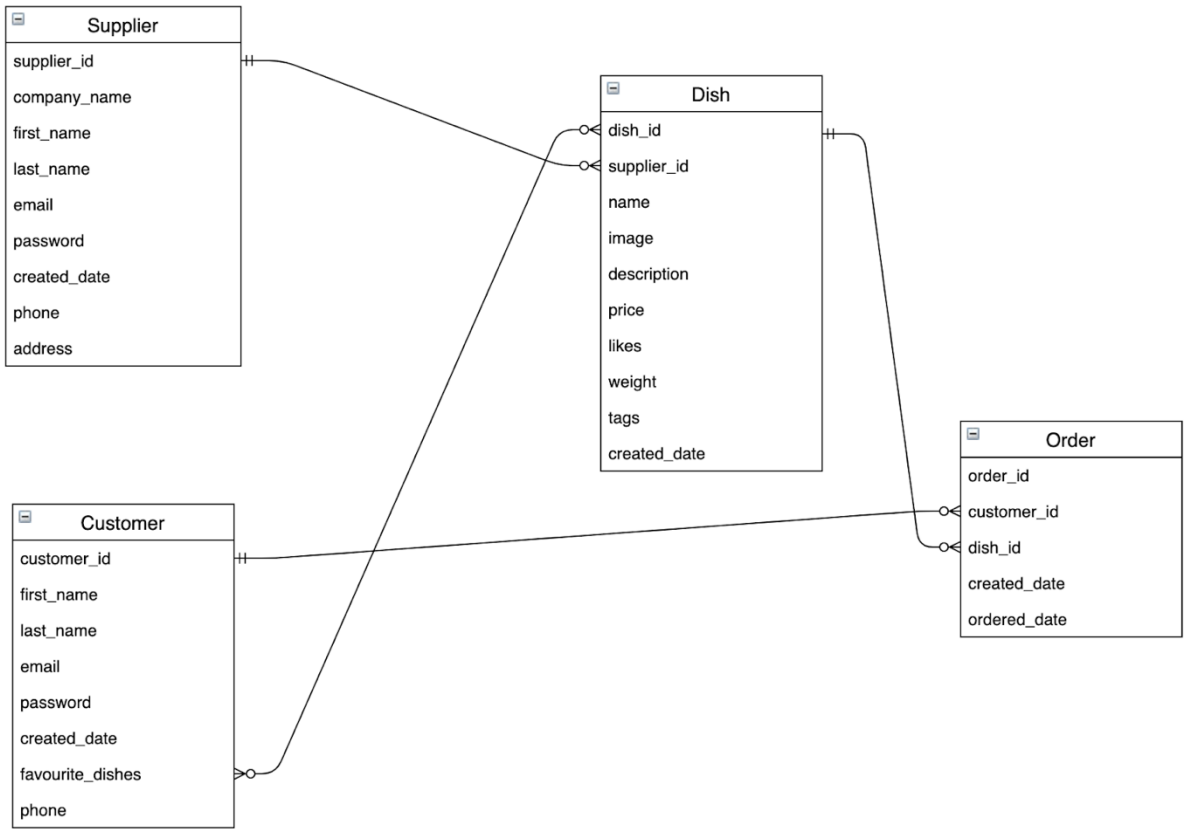


Рис. 3. Структура бази даних

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-СЕРВІСУ

4.1. Особливості реалізації

Відповідно до наведених вище вимог до розробленого веб-додатку була розроблена модульна організація коду, яка сприяє легкому впровадженню нової функціональність до структури вже існуючої в додатку. Основні модулі веб-застосунку знаходяться в серверній частині додатку (рис. 4):

- модуль авторизації;
- модуль генерування звітності;
- модуль валідації даних;
- модуль маршрутизації;
- модуль моделей.

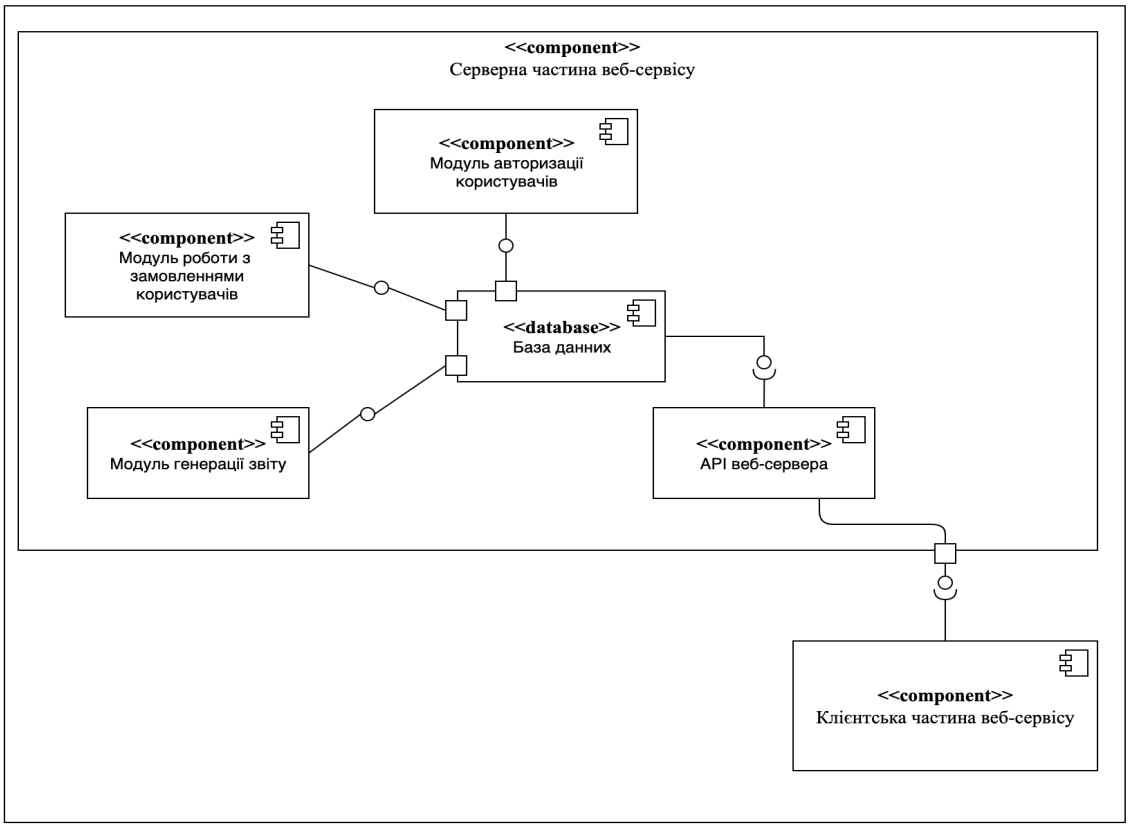


Рис. 4. Компоненти системи

Структура клієнтської частини додатку забезпечена поєднанням засобів Wix Style React, React-компонентів та шаблонів, які розміщені в пакеті client.

4.1.1. Процеси перетворення даних

Процес відображення будь-якої сторінки веб-додатку починається з GET запиту користувача. В свою чергу серверна частина відповідає за обробку цих запитів, так як там знаходяться всі ресурси та бізнес-логіка додатку. Шляхом взаємодії з базою даних, розміщених у модулі моделей додатку, відбувається відбір даних для подальшого представлення. Функції-обробки запитів повертають HTML-сторінки з відібраними даними для рендеренгу їх на клієнтській частині додатку. Схематичне зображення процесу перетворення даних наводиться на рис. 5.

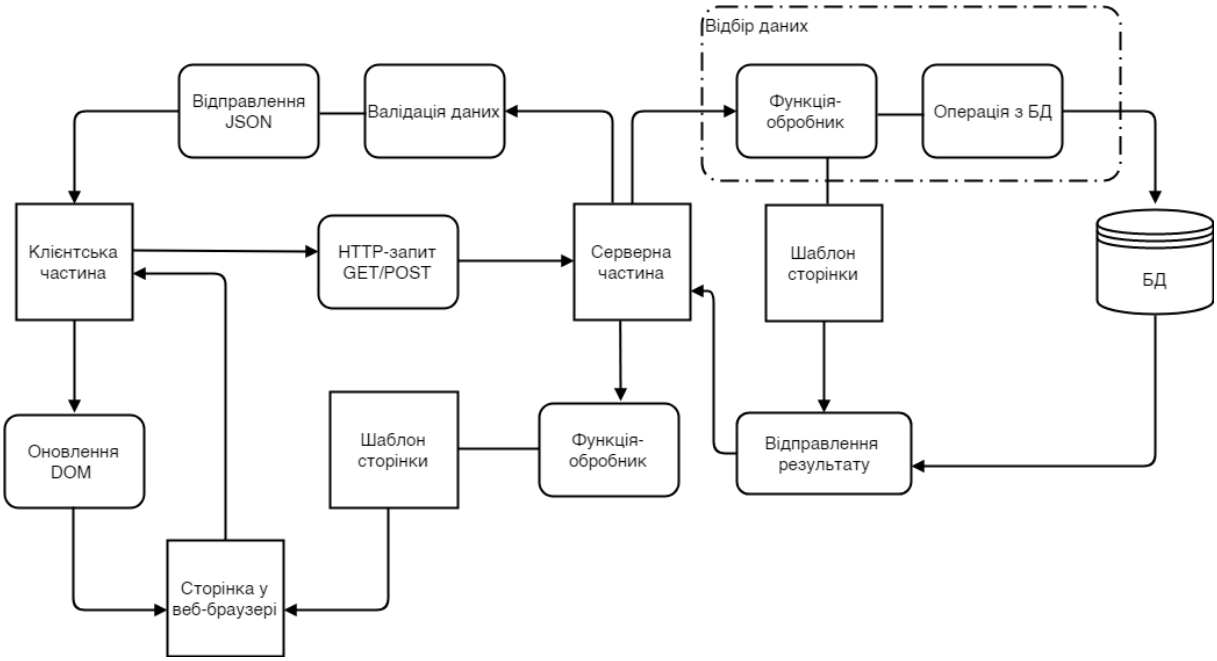


Рис. 5. Процеси перетворення даних в системі

Для будь-якої зміни даних системи, у веб-додатку виконуються тільки POST, DELETE або UPDATE запити, наприклад додавання або видалення кулінарного виробу постачальником. Дані про новий кулінарний виріб проходять валідацію, як на клієнтській частині, так і на серверній

частині веб-застосунку у функції-обробнику. В разі успішності запиту клієнт отримує дані у форматі JSON для оновлення DOM.

4.1.2. Процес авторизації користувачів

Для реалізації механізму авторизації у даному веб-застосунку було використано стандарт JWT (JSON Web Token), що визначає компактний і автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON. За рахунок використання приватного ключу шифрування на етапі створення повідомлення перед відправкою досягається захищеність.

Сам JSON Web Token являє собою три блоки, розділених крапкою: набір полів даних (payload), сигнатура (signature) та заголовок (header). Заголовок та набір полів даних додатково закодовані у формат base64 та представлені у JSON-форматі. Набір полів містить довільні пари ключ-значення. Сигнатура може генеруватися за допомогою як асиметричних, так і симетричних алгоритмів шифрування. Отриманий в результаті токен є засобом авторизації кожного запиту до серверу. Сервер генерує токен, що засвідчує особу користувача, за допомогою приватного ключа, який зберігається на сервері, і надсилає його клієнту. Клієнт відправляє токен назад на сервер при кожному наступному запиті. В свою чергу сервер розшифровує токен завдяки приватному ключу і тим самим розуміє, що запит походить від певної особи (рис. 6).

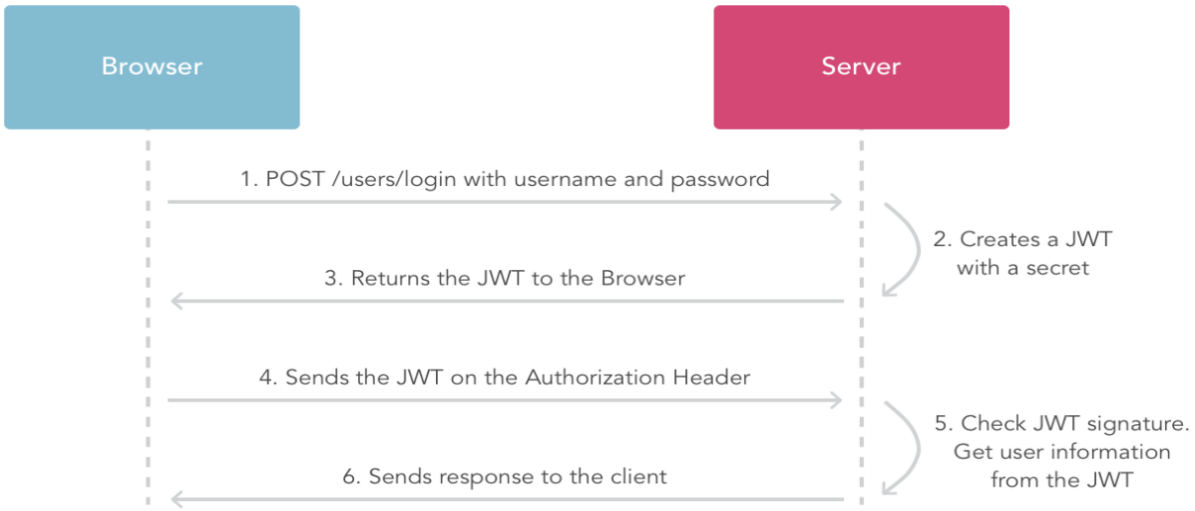


Рис. 6. Схема роботи стандарту JWT

Універсальність стандарту JWT дає змогу взаємодіяти серверу як із веб-додатками, так і з іншими типами: мобільні, десктопні тощо. Приклад імплементації механізму авторизації зображено на лістингу 1.

Лістинг 1. Зчитування JWT токена

```
export const authUser: MiddlewareParams = (req, res, next) => {
  try {
    let token = req.headers.authorization;
    if (token.startsWith('Bearer ')) {
      token = token.slice(7, token.length);
    }

    const decoded = jwt.verify(token, secretKey) as { user: JWTToken };
    if (getUserById(decoded.user._id)) {
      req.decoded = decoded.user;
      next();
    } else {
      respondError(res, 404, 'This user does not exist');
    }
  } catch (e) {
    respondError(res, 403, e);
  }
};
```

4.2. Тестування веб-сервісу

Для забезпечення якості веб-сервісу необхідно провести його тестування відповідно до вимог викладених у підрозділі 3.1. Для перевірки правильності роботи веб-застосунку було обране димове тестування, яке полягає виконанні тест-кейсів, наведених у табл. 9. Результати попереднього тест-кейсу можуть використовуватися у подальших тест-кейсах. Для тестування було використано ОС macOS версії 10.15.4 та браузер Google Chrome версії 75.

Таблиця 9

Тестові випадки

№	Мета тест-кейсу	Опис дій тест-кейсу	Очікуваний результат
1	Перевірити можливість авторизації.	1. Перейти на сторінку авторизації. 2. Ввести логін та пароль. 3. Натиснути кнопку Log in.	1. Система показує сторінку авторизації згідно дизайну. 2. Пароль представлений у вигляді крапок. 3. Користувач переходить на головну сторінку.
2	Перевірити можливість вибору дня наступного тижня.	1. Перейти на головну сторінку. 2. Натиснути на день тижня.	1. Система показує сторінку згідно дизайну. 2. Відображається повний список кулінарних виробів. 3. Обраний день переходить у виділений стан.
3	Перевірити можливість фільтрації страв за постачальником.	1. Натиснути на кнопку обраного постачальника у верхній панелі.	1. Система показує тільки список кулінарних виробів обраного постачальника.
4	Перевірити можливість фільтрації страв за видом страви.	1. Натиснути на поле Select by tags. 2. Обрати вид страви.	1. Відображається список типів страв 2. Система показує тільки список страв тільки певного виду.

Продовження таблиці 9

5	Перевірити можливість вибору страви.	1. Обрати страву з отриманого списку кулінарних виробів.	1. Система показує картинку обраної страви замість дня тижня.
6	Перевірити можливість видалити замовлення з певного дня.	1. Натиснути на іконку кошика, яка знаходиться в хедері. 2. Натиснути на іконку видалення, навпроти обраного дня тижня.	1. Відображається модальне вікно з зробленими замовленнями на наступний тиждень. 2. Дані про замовлення на обраний день видаляються.Іконка дня стає пустою.
7	Перевірити можливість підтвердження замовлення на наступний тиждень	1. Натиснути на іконку кошика, яка знаходиться в хедері. 2. Натиснути на кнопку OK.	1. Відображається модальне вікно з зробленими замовленнями на наступний тиждень. 2. Відображається модальне вікно про успішне підтвердження замовлення.
8	Перевірити можливість виходу з систему.	1. Натиснути на кнопку Logout, яка знаходиться в хедері.	1. Система перенаправляє користувача на сторінку авторизації згідно дизайну.

У табл. 9 наведено основні тест-кейси, які є найнеобхіднішими для повноцінної роботи веб-сервісу для замовлення кулінарних виробів

співробітниками компанії. В результаті перевірки та оцінки якості за допомогою даних тест-кейсів відхилень від вимог не виявлено.

4.3. Порівняння розробки з аналогами

Вимоги до розробленої системи були складені таким чином, щоб задовольнити потреби та критерії, які не враховані існуючими на ринку рішеннями. Підсумуємо переваги:

- реалізація зрозумілого та простого користувацького інтерфейсу;
- можливість додавання власних постачальників;
- динамічне оновлення кулінарних виробів;
- можливість додавання кулінарних виробів до списку улюблених страв;
- можливість аналізу кулінарних виробів та постачальників за кількістю замовлень;
- можливість редагування вже зроблених замовлень;
- можливість фільтрації списку кулінарних виробів за постачальником та за видом страви;
- генерація звіту замовлень у форматі csv;
- авторизація через Google аккаунт;
- доступність з будь-якого пристрою.

Веб-додаток з означеними перевагами може вважатися конкурентним аналогом існуючих рішень.

4.4. Рекомендації щодо подальшого вдосконалення

Модульна організація клієнт-серверної архітектури розробленого веб-додатку надає можливість легко впроваджувати нову функціональність до структури вже існуючої в додатку.

Потенційні напрямки розширення можливостей додатку:

- можливість користувачу залишати коментарі до кожної страви та постачальника;

- можливість додавання кулінарного виробу до списку улюблених страв;
- можливість для постачальника аналізувати замовлення за кількістю та частотою;
- можливість для адміністратора аналізувати постачальників за частотою замовлень у них та за популярністю;
- можливість додавати додаткову інформацію до замовлення;
- інтеграція з мобільними пристроями;
- генерація звітності по кожному співробітнику компанії.

ВИСНОВКИ

Даний дипломний проект присвячено створенню веб-сервісу для замовлення страв співробітниками компанії.

Було зібрано та проаналізовано потреби компаній та їх співробітників у організації харчування на підприємстві. На основі цих потреб сформульовано критерії оцінювання існуючих на ринку програмних рішень. Розглянуті аналоги не задовольняють поставленим вимогам у повній мірі, чим обґрунтовується необхідність розроблення додатку для організації харчування в офісах.

Для розроблення була використана мова програмування JavaScript та відповідні технології: платформа з відкритим кодом Node.js для реалізації серверної частини додатку та бібліотека React з SCSS фреймворком для реалізації інтерфейсу користувачів. Зазначені засоби реалізації дозволили забезпечити стабільну роботу системи.

Розроблене програмне забезпечення забезпечує основні та додаткові функцій з організації харчування:

- дозволяє компаніям додавати власних користувачів;
- дозволяє співробітникам компаній замовляти кулінарні вироби у постачальників;
- дозволяє постачальникам генерувати звіти замовлень;
- дозволяє користувача-співробітникам фільтрувати кулінарні вироби за постачальником та видом страви.

Особливу увагу під час розроблення даного програмного продукту було приділено до інтерфейсу користувачів.

Розроблення виконано у повному обсязі, відповідає всім вимогам, тестування продукту виконано у відповідності до затвердженої програми та методики тестування.

Використання розробленого продукту дозволить компаніям автоматизувати організацію харчування для їх співробітників.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Glassdoor [Електронний ресурс]. — Режим доступу до ресурсу: https://www.glassdoor.com/Award/Best-Places-to-Work-LST_KQ0,19.html
2. Meido [Електронний ресурс]. — Режим доступу до ресурсу: https://getmeido.com/en
3. JSolutions [Електронний ресурс]. — Режим доступу до ресурсу: https://jsolutions.ua/ua/avtomatizatciya-korporativnogo-pitaniya
4. Desktop Application [Електронний ресурс]. — Режим доступу до ресурсу: https://www.pcmag.com/encyclopedia/term/desktop-application
5. Desktop Vs Mobile Vs Web Application [Електронний ресурс]. — Режим доступу до ресурсу: http://www.iomworld.com/desktop-application-vs-mobile-app-vs-web-app-2
6. Реляційна база даних [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85
7. NoSQL vs SQL [Електронний ресурс]. — Режим доступу до ресурсу: https://www.got-it.ai/solutions/sqlquerychat/sql-help/others/nosql-vs-sql-advantages-and-disadvantages.
8. NoSQL [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/NoSQL
9. SQL vs. NoSQL [Електронний ресурс]. — Режим доступу до ресурсу: https://bitnine.net/blog-computing/sql-vs-nosql-comparative-advantages-and-disadvantages.
10. Python [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python
11. Advantages and Disadvantages of Python – How it is dominating programming world [Електронний ресурс]. — Режим доступу до ресурсу:

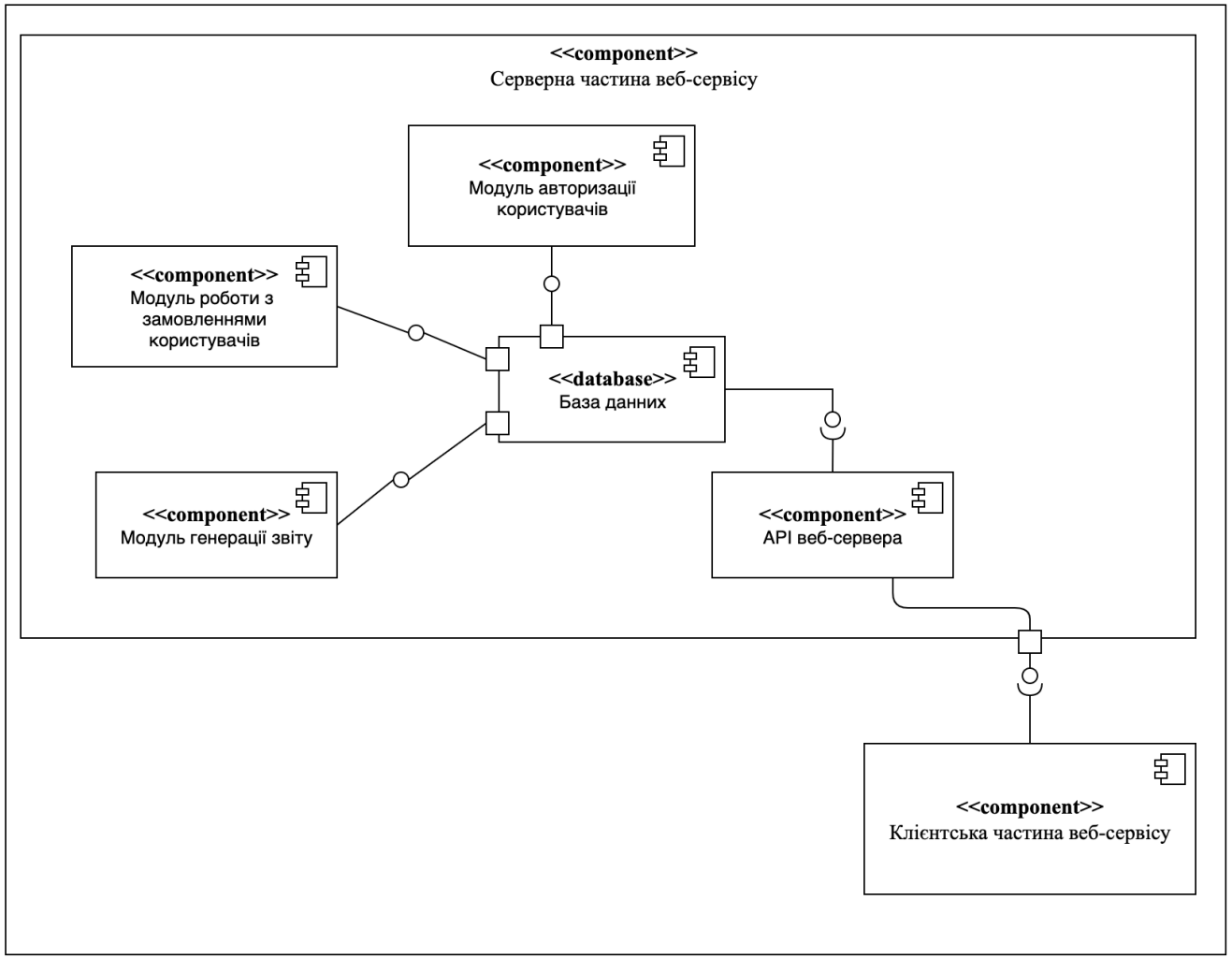
https://techvidvan.com/tutorials/python-advantages-and-disadvantages.

12. Мультипарадигмальна мова програмування [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Мультипарадигмальна_мова_програмування
13. JavaScript MDN [Електронний ресурс]. — Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript
14. JavaScript [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript
15. Vue.js [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Vue.js
16. The Good and the Bad of Vue.js Framework Programming [Електронний ресурс]. — Режим доступу до ресурсу: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js.
17. Angular: [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS
18. What are the Advantages and Disadvantages of Angular [Електронний ресурс]. — Режим доступу до ресурсу: https://www.edureka.co/blog/advantages-and-disadvantages-of-angular.
19. Intro to React [Електронний ресурс]. — Режим доступу до ресурсу: https://reactjs.org/tutorial/tutorial.html
20. React [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/React
21. The Good and the Bad of ReactJS [Електронний ресурс]. — Режим доступу до ресурсу: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native.
22. React, Angular and Vue.js Analysis [Електронний ресурс]. — Режим доступу до ресурсу: https://trends.google.com/trends/explore?cat=31&q=React,Angular,Vue

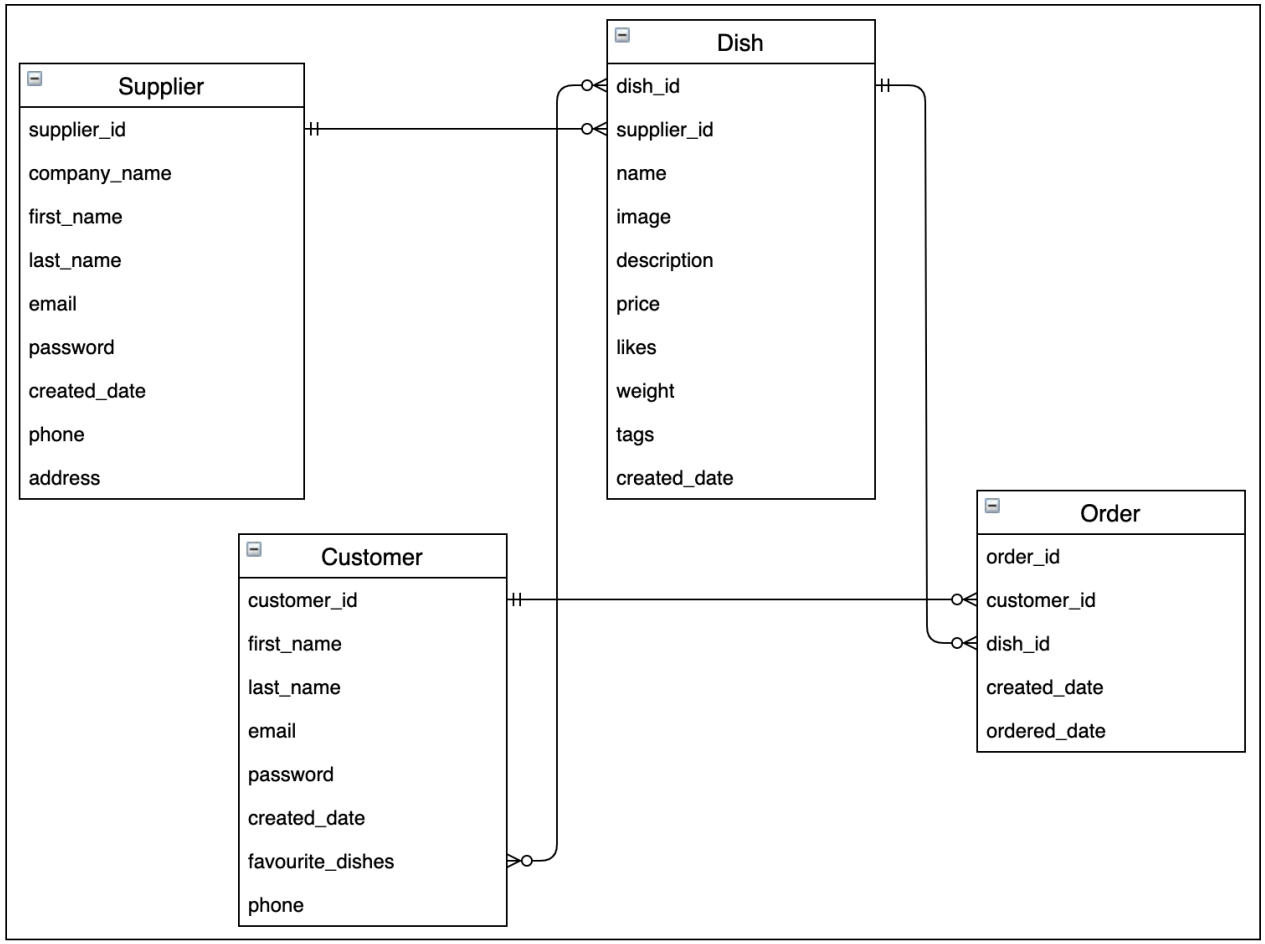
ДОДАТКИ

Додаток 1

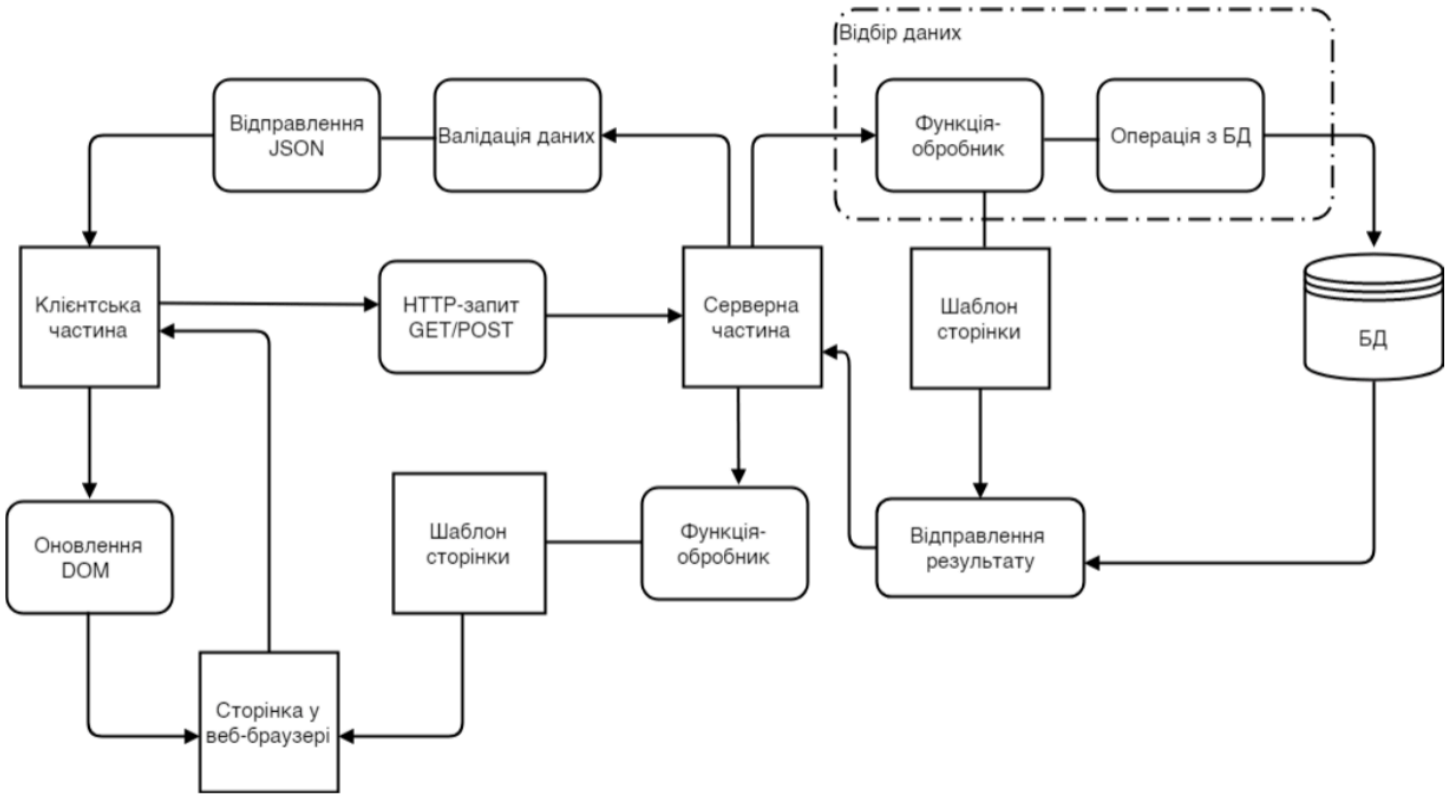
Копії графічних матеріалів



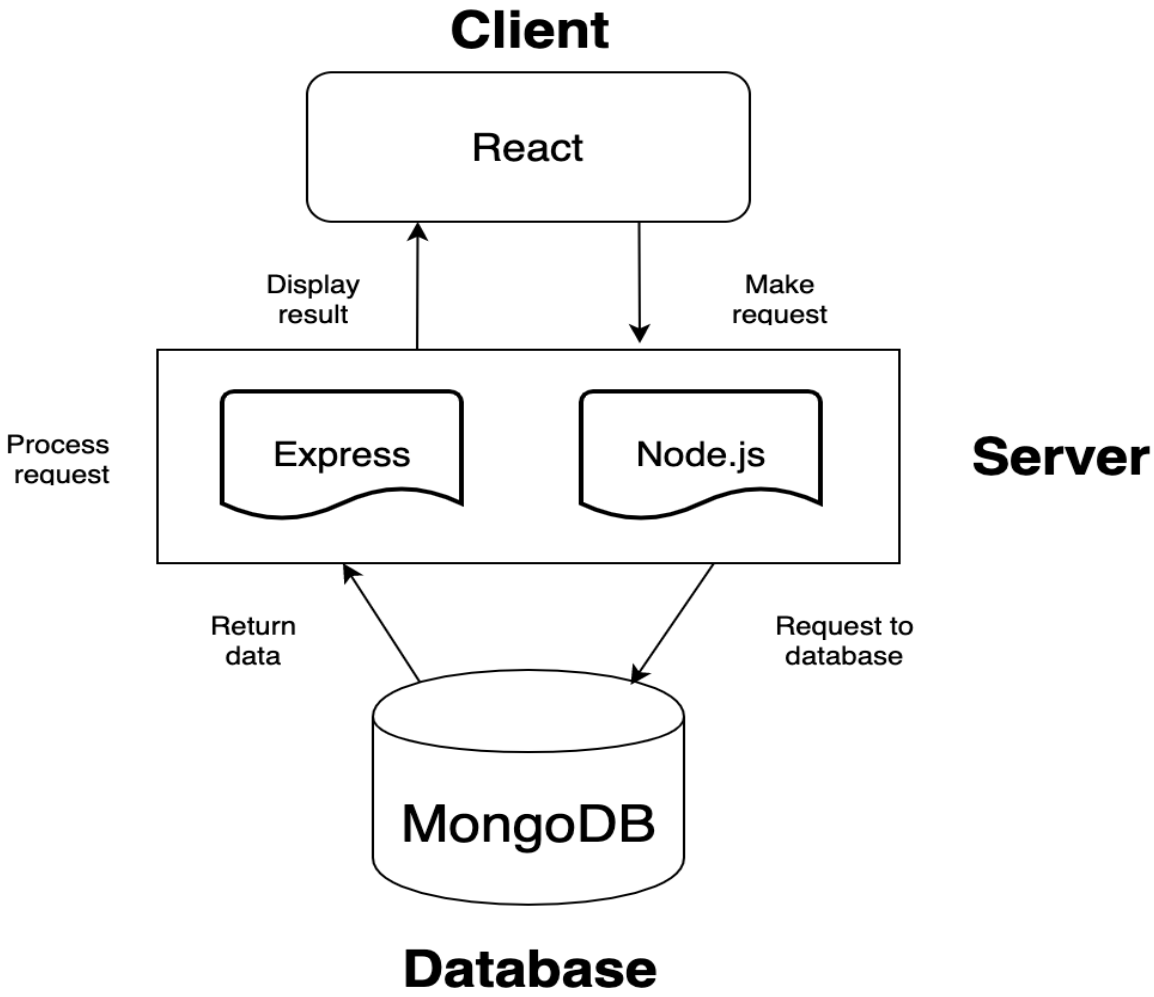
ДП.045440-06-99.
Веб-сервіс для замовлення кулінарних виробів співробітниками компаній. Модулі системи.
UML-діаграма



ДП.045440-07-99.
Веб-сервіс для замовлення страв співробітниками компаній. Схема бази даних системи. ER-діаграма



Дзенік Данило Миколайович
гр. КП-62



Дзенік Данило Миколайович
гр. КП-62

Додаток 2

Текст програми

```
import React from 'react';
import PropTypes from 'prop-types';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import TableSortLabel from '@material-ui/core/TableSortLabel';
import Typography from '@material-ui/core/Typography';
import Paper from '@material-ui/core/Paper';
import IconButton from '@material-ui/core/IconButton';
import KeyboardArrowDownIcon from '@material-ui/icons/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@material-ui/icons/KeyboardArrowUp';
import Collapse from "@material-ui/core/Collapse/Collapse";
import Box from "@material-ui/core/Box";


function descendingComparator(a, b, orderBy) {
    if (b[orderBy] < a[orderBy]) {
        return -1;
    }
    if (b[orderBy] > a[orderBy]) {
        return 1;
    }
    return 0;
}

function getComparator(order, orderBy) {
    return order === 'desc'
        ? (a, b) => descendingComparator(a, b, orderBy)
        : (a, b) => -descendingComparator(a, b, orderBy);
}

function stableSort(array, comparator) {
    const stabilizedThis = array.map((el, index) => [el, index]);
    stabilizedThis.sort((a, b) => {
        const order = comparator(a[0], b[0]);
        if (order !== 0) return order;
        return a[1] - b[1];
    });
    return stabilizedThis.map((el) => el[0]);
}

const headEventCells = [
    { id: 'name', numeric: false, disablePadding: true, label: 'Event Name'
},
    { id: 'rate', numeric: true, disablePadding: false, label: 'Event Rate'
},
    { id: 'start', numeric: true, disablePadding: false, label: 'Start Date'
},
    { id: 'marksCount', numeric: true, disablePadding: false, label: 'Count
of time tracks' },
];
const headCategoriesCells = [
    { id: 'name', numeric: false, disablePadding: true, label: 'Category
Name' },
    { id: 'rate', numeric: true, disablePadding: false, label: 'Category
Rate' },
    { id: 'eventsCount', numeric: true, disablePadding: false, label: 'Count
of Category events' },
];
const headStudentCells = [
```

```
    { id: 'name', numeric: false, disablePadding: true, label: 'Student Name'
},
    { id: 'rate', numeric: true, disablePadding: false, label: 'Student Rate'
},
    { id: 'time', numeric: true, disablePadding: false, label: 'Student
Approved Time' },
    { id: 'marksCount', numeric: true, disablePadding: false, label: 'Count
of time tracks' },
];

function EnhancedTableHead(props) {
    const { classes, order, orderBy, onRequestSort, tableType } = props;
    const createSortHandler = (property) => (event) => {
        onRequestSort(event, property);
    };
    let headCells = [];
    switch (tableType) {
        case 'events':
            headCells = headEventCells;
            break;
        case 'categories':
            headCells = headCategoriesCells;
            break;
        case 'students':
            headCells = headStudentCells;
            break;
    }

    return (
        <TableHead>
            <TableRow>
                <TableCell padding="checkbox">
                </TableCell>
                {headCells.map((headCell) => (
                    <TableCell
                        key={headCell.id}
                        align={headCell.numeric ? 'right' : 'left'}
                        padding={headCell.disablePadding ? 'none' :
'default'}
                        sortDirection={orderBy === headCell.id ? order :
false}
                    >
                        <TableSortLabel
                            active={orderBy === headCell.id}
                            direction={orderBy === headCell.id ? order :
'asc'}
                            onClick={createSortHandler(headCell.id)}
                        >
                            {headCell.label}
                            {orderBy === headCell.id ? (
                                <span className={classes.visuallyHidden}>
                  {order === 'desc' ? 'sorted descending' : 'sorted
ascending'}
                </span>
                            ) : null}
                        </TableSortLabel>
                    </TableCell>
                ))}
            </TableRow>
        </TableHead>
    );
}

EnhancedTableHead.propTypes = {
```

```
    classes: PropTypes.object.isRequired,
    numSelected: PropTypes.number.isRequired,
    onRequestSort: PropTypes.func.isRequired,
    onSelectAllClick: PropTypes.func.isRequired,
    order: PropTypes.oneOf(['asc', 'desc']).isRequired,
    orderBy: PropTypes.string.isRequired,
    rowCount: PropTypes.number.isRequired,
};

const useStyles = makeStyles((theme) => ({
    root: {
        width: '100%',
    },
    paper: {
        width: '100%',
        marginBottom: theme.spacing(2),
    },
    table: {
        minWidth: 750,
    },
    visuallyHidden: {
        border: 0,
        clip: 'rect(0 0 0 0)',
        height: 1,
        margin: -1,
        overflow: 'hidden',
        padding: 0,
        position: 'absolute',
        top: 20,
        width: 1,
    },
}));


function EventTableBody({rows, isSelected, order, orderBy}) {
    const [open, setOpen] = React.useState({});
    return(
        <TableBody>
            {stableSort(rows, getComparator(order, orderBy))
                .map((row, index) => {
                    const isItemSelected = isSelected(row.name);
                    const labelId = `enhanced-table-checkbox-${index}`;

                    return (
                        <React.Fragment key={index}>
                            <TableRow
                                hover
                                aria-checked={isItemSelected}
                                tabIndex={-1}
                                key={row.name}
                                selected={isItemSelected}
                            >
                                <TableCell component="th" scope="row">
                                    <IconButton aria-label="expand row"
size="small" disabled={row.marksCount === 0} onClick={() => {
                                        let newOpen = {...open};
                                        newOpen[`${index}`] =
!newOpen[`${index}`];
                                        setOpen({...newOpen})
                                    }}>
                                        {open[`${index}`] ?
<KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
                                    </IconButton>
                                </TableCell>
```

```
                                    <TableCell component="th" id={labelId}
scope="row" padding="none">
                                        {row.name}
                                    </TableCell>
                                    <TableCell
align="right">{row.rate}</TableCell>
                                    <TableCell align="right">{new
Date(row.start).toDateString()}</TableCell>
                                    <TableCell
align="right">{row.marksCount}</TableCell>
                                </TableRow>
                                <TableCell style={{ paddingBottom: 0, paddingTop:
0 }} colSpan={6}>
                                    <Collapse in={open[`${index}`]}
timeout="auto" unmountOnExit>
                                        <Box margin={1}>
                                            <Typography variant="h6" gutterBottom
component="div">
                                                Time Marks
                                            </Typography>
                                            <Table size="small" aria-
label="purchases">
                                                <TableHead>
                                                    <TableRow>
                                                        <TableCell>Student
Name</TableCell>

<TableCell>Rate</TableCell>

<TableCell>Feedback</TableCell>
                                                    </TableRow>
                                                </TableHead>
                                                <TableBody>
                                                    {row.marks.map((item,
index)=>{

                                                        return(
                                                            <TableRow
key={index}>

                                                            <TableCell
component="th" scope="row">
                                                                {item.name}
                                                            </TableCell>

<TableCell>{item.rate}</TableCell>

<TableCell>{item.feedback}</TableCell>
                                                            </TableRow>
                                                        )
                                                        })}
                                                </TableBody>
                                            </Table>
                                        </Box>
                                    </Collapse>
                                </TableCell>

                            </React.Fragment>
                        );
                    })}
            </TableBody>
        )
}

function CategoryTableBody({rows, isSelected, order, orderBy}) {
    const [open, setOpen] = React.useState({});
```

```
    return(
        <TableBody>
            {stableSort(rows, getComparator(order, orderBy))
                .map((row, index) => {
                    const isItemSelected = isSelected(row.name);
                    const labelId = `enhanced-table-checkbox-${index}`;

                    return (
                        <React.Fragment key={index}>
                            <TableRow
                                hover
                                aria-checked={isItemSelected}
                                tabIndex={-1}
                                key={row.name}
                                selected={isItemSelected}
                            >
                                <TableCell component="th" scope="row">
                                    <IconButton aria-label="expand row"
size="small" disabled={row.eventsCount === 0} onClick={() => {
                                        let newOpen = {...open};
                                        newOpen[`${index}`] =
!newOpen[`${index}`];
                                        setOpen({...newOpen})
                                    }}>
                                        {open[`${index}`] ?
<KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
                                    </IconButton>
                                </TableCell>
                                <TableCell component="th" id={labelId}
scope="row" padding="none">
                                    {row.name}
                                </TableCell>
                                <TableCell
align="right">{row.rate}</TableCell>
                                <TableCell
align="right">{row.eventsCount}</TableCell>
                            </TableRow>
                            <TableCell style={{ paddingBottom: 0, paddingTop:
0 }} colSpan={6}>
                                <Collapse in={open[`${index}`]}
timeout="auto" unmountOnExit>
                                    <Box margin={1}>
                                        <Typography variant="h6" gutterBottom
component="div">
                                            Events
                                        </Typography>
                                        <Table size="small" aria-
label="purchases">
                                            <TableHead>
                                                <TableRow>
                                                    <TableCell>Event
Name</TableCell>

<TableCell>Rate</TableCell>
                                                    <TableCell>Start
Date</TableCell>
                                                </TableRow>
                                            </TableHead>
                                            <TableBody>

{row.categoryEvents.map((item, index)=>{
                                                    return(
                                                        <TableRow
key={index}>
```

```
                                                                <TableCell
component="th" scope="row">
                                                                    {item.name}
                                                                </TableCell>

<TableCell>{item.rate}</TableCell>
                                                                <TableCell>{new
Date(item.start).toDateString()}</TableCell>
                                                            </TableRow>
                                                        )
                                                    })}
                                                </TableBody>
                                            </Table>
                                        </Box>
                                    </Collapse>
                                </TableCell>

                            </React.Fragment>
                        );
                    })}
            </TableBody>
        )
}


function StudentTableBody({rows, isSelected, order, orderBy}) {
    const [open, setOpen] = React.useState({});
    return(
        <TableBody>
            {stableSort(rows, getComparator(order, orderBy))
                .map((row, index) => {
                    const isItemSelected = isSelected(row.name);
                    const labelId = `enhanced-table-checkbox-${index}`;

                    return (
                        <React.Fragment key={index}>
                            <TableRow
                                hover
                                aria-checked={isItemSelected}
                                tabIndex={-1}
                                key={row.name}
                                selected={isItemSelected}
                            >
                                <TableCell component="th" scope="row">
                                    <IconButton aria-label="expand row"
size="small" disabled={row.marksCount === 0} onClick={() => {
                                        let newOpen = {...open};
                                        newOpen[`${index}`] =
!newOpen[`${index}`];
                                        setOpen({...newOpen})
                                    }}>
                                        {open[`${index}`] ?
<KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
                                    </IconButton>
                                </TableCell>
                                <TableCell component="th" id={labelId}
scope="row" padding="none">
                                    {row.name}
                                </TableCell>
                                <TableCell
align="right">{row.rate}</TableCell>
                                <TableCell
align="right">{row.time}</TableCell>
```

```
                                    <TableCell
align="right">{row.marksCount}</TableCell>
                                </TableRow>
                                <TableCell style={{ paddingBottom: 0, paddingTop:
0 }} colSpan={6}>
                                    <Collapse in={open[`${index}`]}
timeout="auto" unmountOnExit>
                                        <Box margin={1}>
                                            <Typography variant="h6" gutterBottom
component="div">
                                                Events
                                            </Typography>
                                            <Table size="small" aria-
label="purchases">
                                                <TableHead>
                                                    <TableRow>
                                                        <TableCell>Event
Name</TableCell>

<TableCell>Rate</TableCell>

<TableCell>Time</TableCell>

<TableCell>Feedback</TableCell>

<TableCell>Status</TableCell>
                                                    </TableRow>
                                                </TableHead>
                                                <TableBody>
                                                    {row.marks.map((item,
index)=>{
                                                        return(
                                                            <TableRow
key={index}>
                                                                <TableCell
component="th" scope="row">

{item.eventName}
                                                                </TableCell>

<TableCell>{item.rate}</TableCell>

<TableCell>{item.time}</TableCell>

<TableCell>{item.feedback}</TableCell>

<TableCell>{item.status}</TableCell>
                                                            </TableRow>
                                                        )
                                                    })}
                                                </TableBody>
                                            </Table>
                                        </Box>
                                    </Collapse>
                                </TableCell>

                            </React.Fragment>
                        );
                    })}
                </TableBody>
            )
}

export default function EnhancedTable({type, rows}) {
```

```
    const classes = useStyles();
    const [order, setOrder] = React.useState('asc');
    const [orderBy, setOrderBy] = React.useState('calories');
    const [selected, setSelected] = React.useState([]);

    const handleRequestSort = (event, property) => {
        const isAsc = orderBy === property && order === 'asc';
        setOrder(isAsc ? 'desc' : 'asc');
        setOrderBy(property);
    };

    const handleSelectAllClick = (event) => {
        if (event.target.checked) {
            const newSelecteds = rows.map((n) => n.name);
            setSelected(newSelecteds);
            return;
        }
        setSelected([]);
    };

    const isSelected = (name) => selected.indexOf(name) !== -1;

    let body = null;
    switch (type) {
        case 'events':
            body = (<EventTableBody rows={rows} isSelected={isSelected}
order={order} orderBy={orderBy}/>);
            break;
        case 'categories':
            body = (<CategoryTableBody rows={rows} isSelected={isSelected}
order={order} orderBy={orderBy}/>);
            break;
        case 'students':
            body = (<StudentTableBody rows={rows} isSelected={isSelected}
order={order} orderBy={orderBy}/>);
            break;

    }

    return (
        <div className={classes.root}>
            <Paper className={classes.paper}>
                <TableContainer>
                    <Table
                        className={classes.table}
                        aria-labelledby="tableTitle"
                        size={'medium'}
                        aria-label="enhanced table"
                    >
                        <EnhancedTableHead
                            tableType={type}
                            classes={classes}
                            numSelected={selected.length}
                            order={order}
                            orderBy={orderBy}
                            onSelectAllClick={handleSelectAllClick}
                            onRequestSort={handleRequestSort}
                            rowCount={rows.length}
                        />
                        {body}
                    </Table>
                </TableContainer>
            </Paper>
        </div>
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Вебсервіс для замовлення кулінарних виробів співробітниками компаній

Виконав: Дзенік Данило Миколайович

Науковий керівник: ст. викл., к.т.н. Рибачок Наталія Антонівна

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: автоматизувати процес замовлення кулінарних виробів співробітниками компанії

Завдання:

1. Вивчити предметну область, проаналізувати існуючі рішення, сформулювати вимоги до системи.
2. Обрати засоби реалізації ПЗ, спроектувати архітектуру системи та структуру БД.
3. Розробити програмне забезпечення.
4. Протестувати та проаналізувати розроблене програмне забезпечення.

АКТУАЛЬНІСТЬ

- Корпоративне харчування стрімко набирає обертів з кожним роком у зв’язку з ефективності покращення умов харчування співробітникам компаній. Близько 60% IT-компаній списку Forbes Global 2000 надають корпоративне харчування як додаткове благо.
- Ручний спосіб організації харчування в офісах займає багато часу та ресурсів.

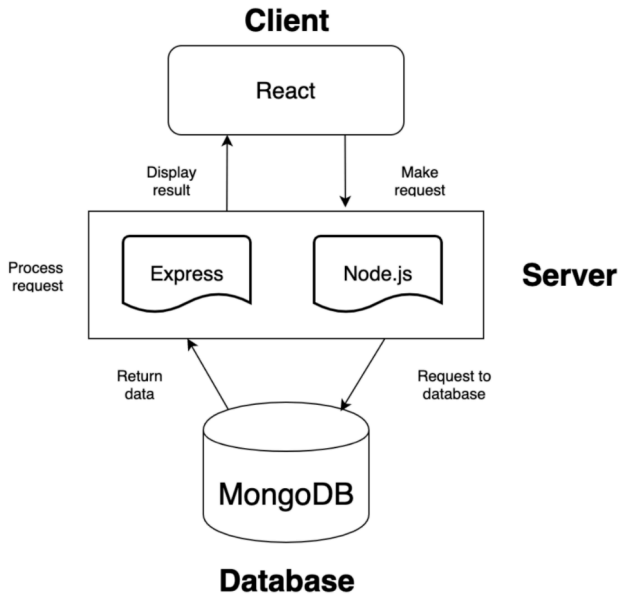
ОСНОВНІ ФУНКЦІОНАЛЬНІ ВИМОГИ

1. Необхідність авторизації для роботи з вебдодатком
2. Користувач-співробітник
   - Перегляд та фільтрація кулінарних виробів.
   - Замовлення кулінарних виробів не раніше наступного тижня.
   - Редагування списку улюблених страв.
   - Редагування вже зроблених замовлень
3. Постачальник
   - Редагування списку кулінарних виробів.
   - Перегляд списку замовлень користувачів.
   - Генерація звіту замовлень у форматі CSV
4. Адміністратор
   - Реєстрація постачальників.
   - Реєстрація співробітників.
   - Перегляд списку замовлень користувачів

НЕФУНКЦІОНАЛЬНІ ВИМОГИ

1. Підтримка роботи додатку в найбільш поширених ОС: ...
2. Зручність, зрозумілість та простота інтерфейсу користувача
3. Наявність англійської та української локалізації

АРХІТЕКТУРА СИСТЕМИ

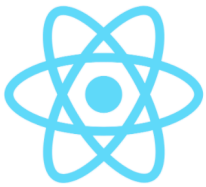


ЗАСІБ РЕАЛІЗАЦІЇ

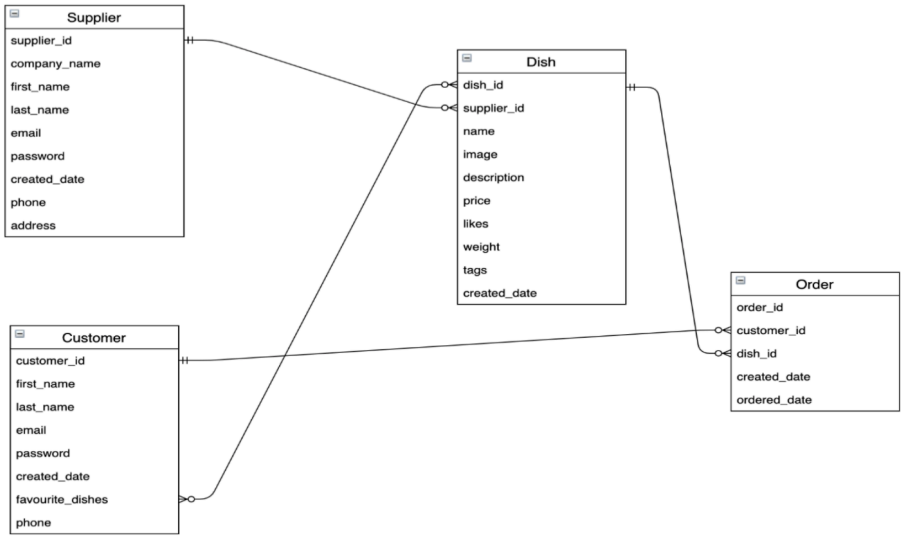
Для реалізації системи обрано форму вебдодатку з огляду на:

- незалежність від цільової платформи;
- легкість встановлення, підтримки та обслуговування;
- можливість доступу користувачів із будь-якого пристрою з підключенням до мережі Інтернет;
- ширші можливості в реалізації інтерфейсу.

ОБРАНІ ЗАСОБИ РЕАЛІЗАЦІЇ



СТРУКТУРА БАЗА ДАНИХ



ПРИКЛАД МОДУЛЯ АВТОРИЗАЦІЇ КОРИСТУВАЧА

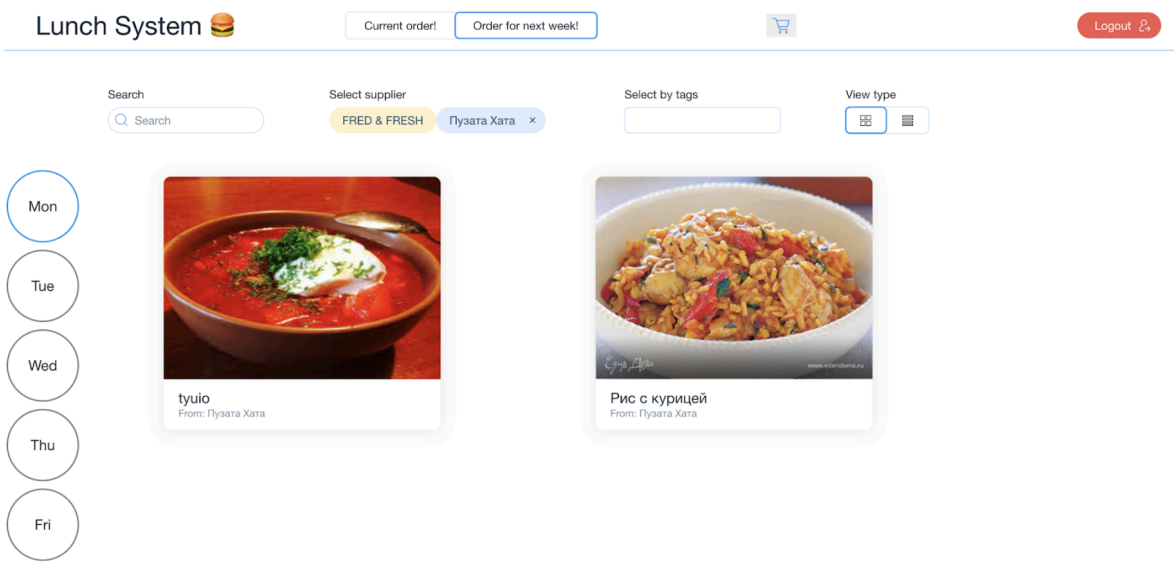

```
export const authUser: MiddlewareParams = (req : ExtendedRequest , res : Response , next : NextFunction ) => {
  try {
    let token = req.headers.authorization;
    if (token.startsWith('Bearer ')) {
      token = token.slice(7, token.length);
    }

    const decoded = jwt.verify(token, secretKey) as { user: JWTToken };
    if (getUserById(decoded.user._id)) {
      req.decoded = decoded.user;
      next();
    } else {
      respondError(res,  statusCode: 404,  message: 'This user does not exist');
    }
  } catch (e) {
    respondError(res,  statusCode: 403, e);
  }
};
```

ПРИКЛАД МОДУЛЯ ГЕНЕРАЦІЇ ЗВІТУ


```
function generateCSVList(selectedIds, orders) {
  const mappedOrders = selectedIds.map(id => {
    const order = orders[id];
    return {
      user: order.userId.name,
      dish: this.getDishNameById(order.dishId),
      floor: order.userId.floor,
      date: moment.utc(order.orderDate).format( format: 'L'),
      comment: order.comment,
    };
  });
  csvExporter.generateCsv(mappedOrders);
}
```

ПРИКЛАД СТОРІНКИ КОРИСТУВАЧА



ТЕСТУВАННЯ

- Ручне тестування;
- Створення стресових ситуацій;
- Димове тестування.

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Деякі з сценаріїв тестування:

- Перевірено авторизацію.
- Перевірено додавання користувачів та постачальників.
- Перевірено додавання нових кулінарних виробів.
- Перевірено створення замовлення користувачем
- Перевірено генерацію звіту постачальником

ПОРІВНЯННЯ З ІСНУЮЧИМИ АНАЛОГАМИ

Функціональні можливості	Розроблений вебсервіс	Meido	JSolutions
Додавання постачальників	+	+	-
Фільтрація страв користувачем	+	-	+
Можливість постачальнику генерувати звіти	+	-	-
Додавання користувачем страви до списку улюблених	+	-	+
Відображення списку кулінарних виробів у форматі галереї або списку	+	_	+
Реєстрація користувачів через адміністратора	+	+	-

РЕКОМЕНДАЦІЇ ЩОДО ВДОСКОНАЛЕННЯ

- Додати можливість онлайн-оплати для адміністратора.
- Додати можливість аналізу замовлень користувачів для постачальника та адміністратора.
- Додати можливість користувачу оцінювати страву.
- Додати можливість користувачу залишати коментарі до кожної страви та постачальника в цілому.
- Інтегрувати з мобільними пристроями.

ВИСНОВКИ

1. Огляд існуючих рішень для автоматизації процесу замовлення кулінарних виробів співробітниками компанії підтвердив актуальність теми дипломного проекту.
2. На основі сформульованих вимог було спроектовано архітектуру вебдодатку, структуру БД.
3. На основі обраних для реалізації сучасних технологій виконано розроблення алгоритмів та функцій ро
4. Тестування виявило / невиявило ….
5. Порівняння розробки з аналогами показало переваги за встановленими критеріями.
6. Визначено напрямки розвитку, які розширять функції та можливості програмного продукту.

ПЕРЕВІРКА НА УНІКАЛЬНІСТЬ

Розділ	Унікальність
1	76%
2	72%
3	90%
4	82%
Диплом повністю:	76%

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ ЗАМОВЛЕННЯ КУЛІНАРНИХ ВИРОБІВ СПІВРОБІТНИКАМИ КОМПАНІЙ

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Данило ДЗЕНІК

2019

ЗМІСТ

1. Об’єкт випробувань 3
2. Мета тестування 3
3. Методи тестування 3
4. Засоби та порядок тестування 4

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-сервіс для замовлення кулінарних виробів співробітниками компаній, який являє собою веб-сайт, серверна частина якого розроблена з використанням фреймворку Express на основі платформи Node.js, а клієнтська – з використанням фреймворку React.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) відповідність функціональних можливостей, реалізованих у програмному застосунку до заявлених;
2) коректність поведінки програмного застосунку;
3) забезпечення належного рівня безпеки даних;
4) зручність роботи з веб-сервісом.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Black Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність платформи перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування веб-ресурсу в різних веб-браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-СЕРВІС ДЛЯ ЗАМОВЛЕННЯ КУЛІНАРНИХ ВИРОБІВ СПІВРОБІТНИКАМИ КОМПАНІЇ

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Наталія РИБАЧОК

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Данило Дзенік

2020

ЗМІСТ

1. Опис структури веб-сервісу 3
2. Процедура авторизації користувача 4
3. Процедура замовлення кулінарного виробу користувачем 5
4. Особиста сторінка користувача 7
5. Процедура додавання користувача у систему 8
6. Процедура редагування кулінарного виробу 10

1. Опис структури веб-сервісу

Веб-сервіс для замовлення кулінарних виробів співробітниками компаній складається з наступних сторінок:

- сторінка авторизації;
- головна сторінка користувача;
- особиста сторінка користувача;
- сторінка додавання користувачів веб-додатку;
- сторінка редагування кулінарних виробів.

Перехід між сторінками відбувається за натисканням відповідних кнопок.

2. Процедура авторизації користувача

Сторінка авторизації користувача відкривається за змовчуванням для неавторизованих користувачів (рис. 1). Сторінка містить форму для вводу необхідних для авторизації даних: юзернейма та пароля. Після введення коректних даних та натиснення кнопки «Login», користувач увійде в систему.

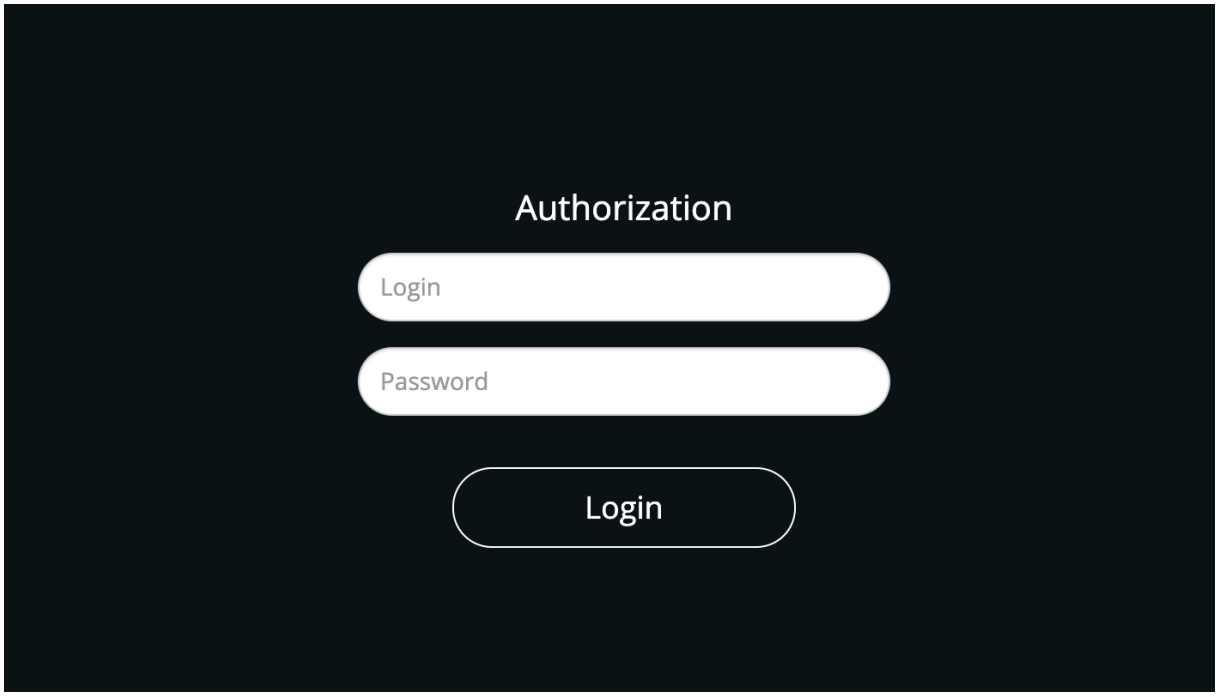


Рис. 1. Сторінка авторизації користувача

3. Процедура замовлення кулінарного виробу користувачем

Після успішної авторизації, користувач потрапляє на головну сторінку, яка містить панель навігації, панель фільтрації списку кулінарних виробів та панель відображення кулінарних виробів. Панель навігації містить в собі кнопки для швидкого переходу на особисту сторінку користувача та головну сторінку веб-додатку,а також кнопку для виходу з системи. Панель фільтрації містить в собі поле для пошуку кулінарних виробів, кнопки для вибору постачальника та тип відображення страв. В даному веб-застосунку присутні два типи відображення списку страв: галерея (рис. 2) та список (рис.3).

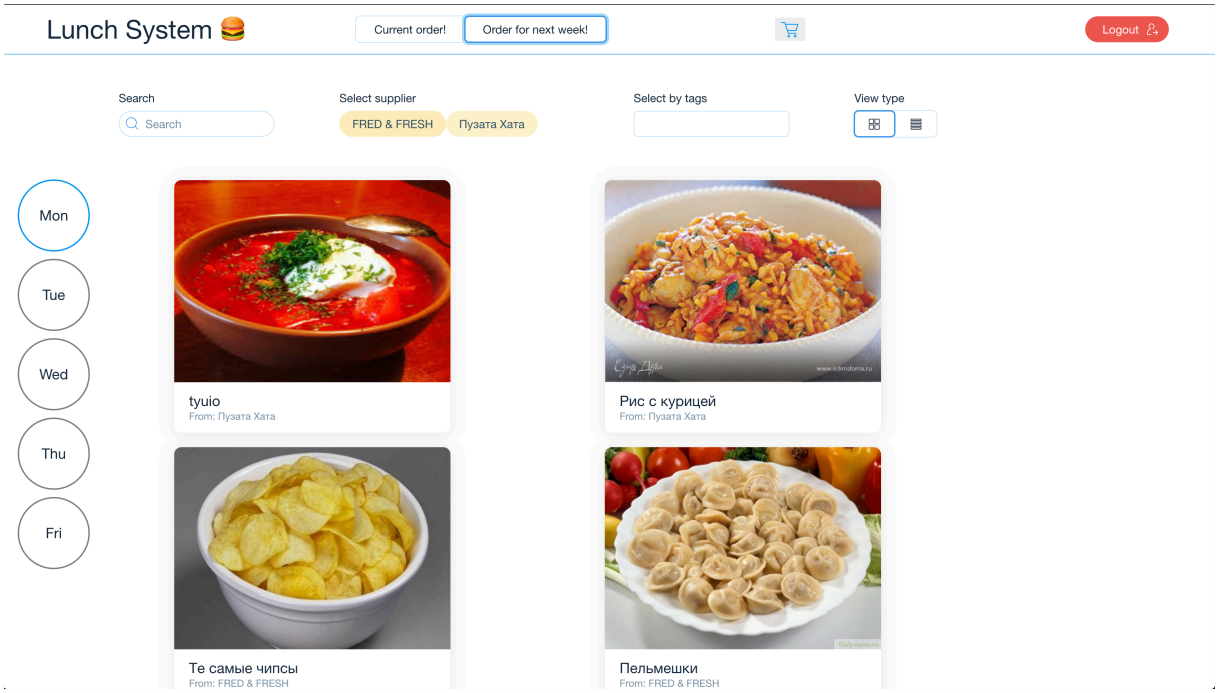


Рис. 2. Головна сторінка користувача з типом відображення галерея

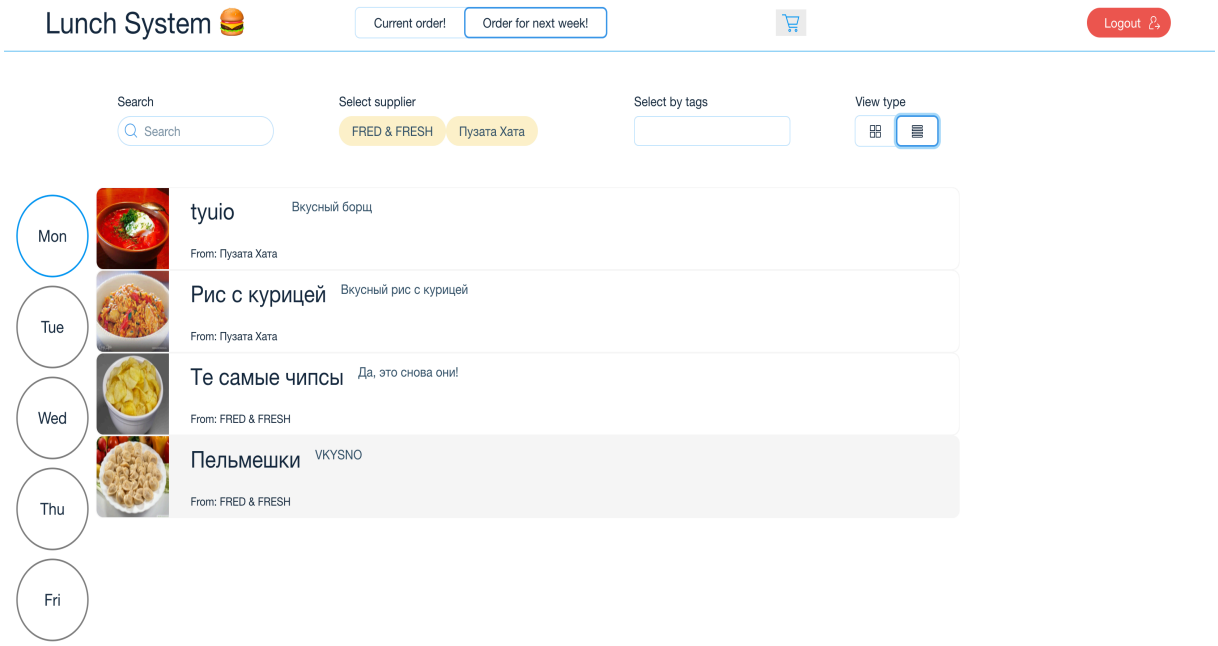


Рис. 3. Головна сторінка користувача з типом відображення список

Основна панель відображення списку страв містить в собі кнопки для вибору дня тижня та коротку інформацію по кожному кулінарному виробу: назва, опис, постачальника та картинку страви. Після вибору страви на кожен день тижня з'являється вікно для підтвердження замовлення (рис. 4).

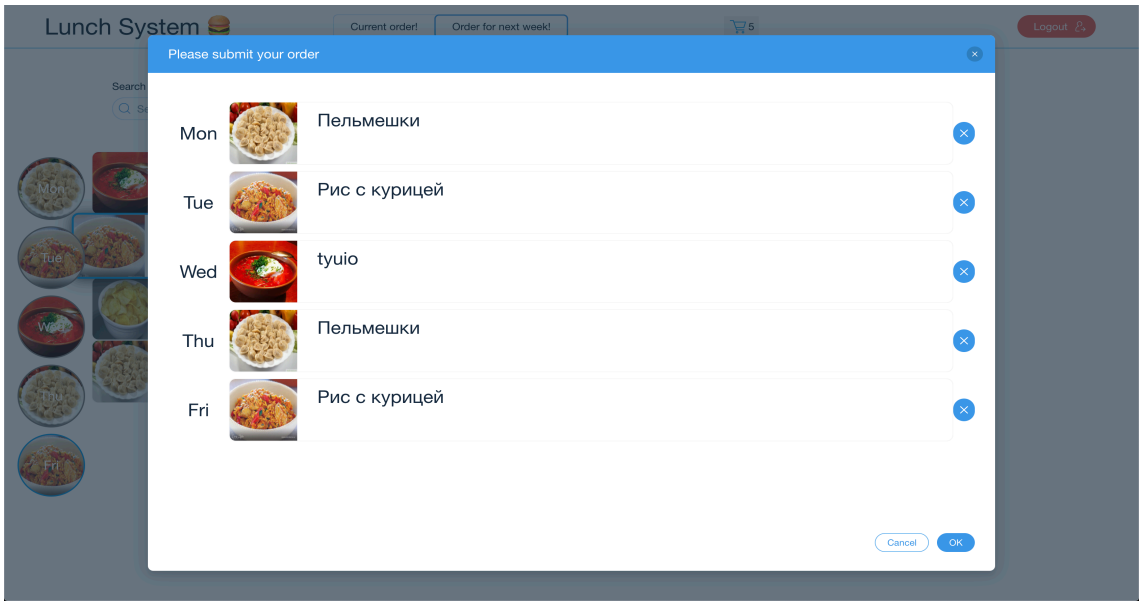


Рис. 4. Головна сторінка користувача з типом відображення галерея

4. Особиста сторінка користувача

Після успішного замовлення страв, користувач потрапляє на особисту сторінку (рис. 5). На цій сторінці містяться інформація про його замовлення на тиждень.

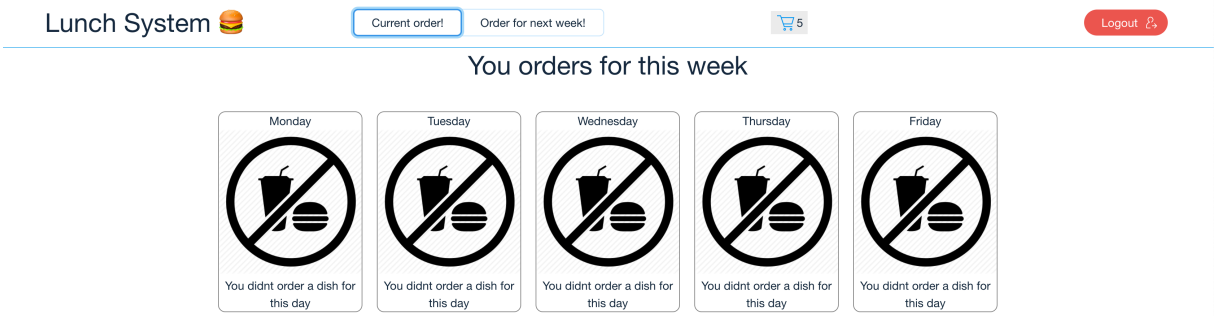


Рис. 5. Особиста сторінка користувача

5. Процедура додавання користувачів веб-додатку

Після успішної авторизації, адміністратор веб-сервісу автоматично потрапляє на особисту сторінку (рис. 6), яка містить таблицю для панель навігації, кнопки для вибору типу користувача, якого необхідно додати, форма для додавання користувача (рис. 7) та таблиця для відображення вже списку існуючих користувачів (рис. 8). Після натискання на кнопку «Add» відбувається перевірка введених даних та додавання нового користувача у список існуючих.

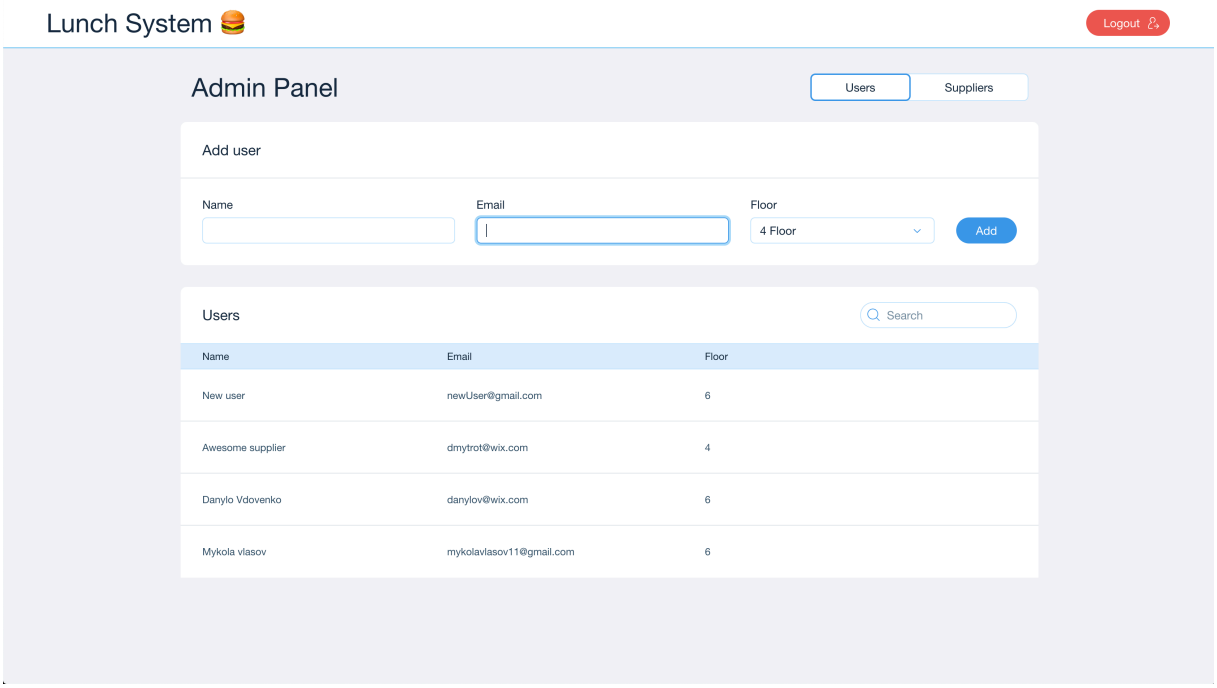


Рис. 6. Сторінка додавання користувачів веб-додатку

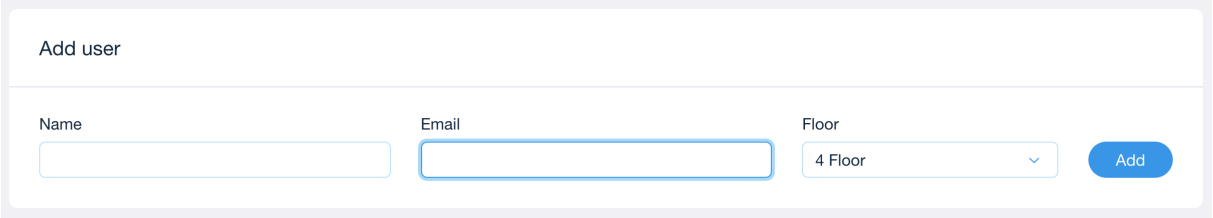


Рис. 7. Форма додавання нового користувача

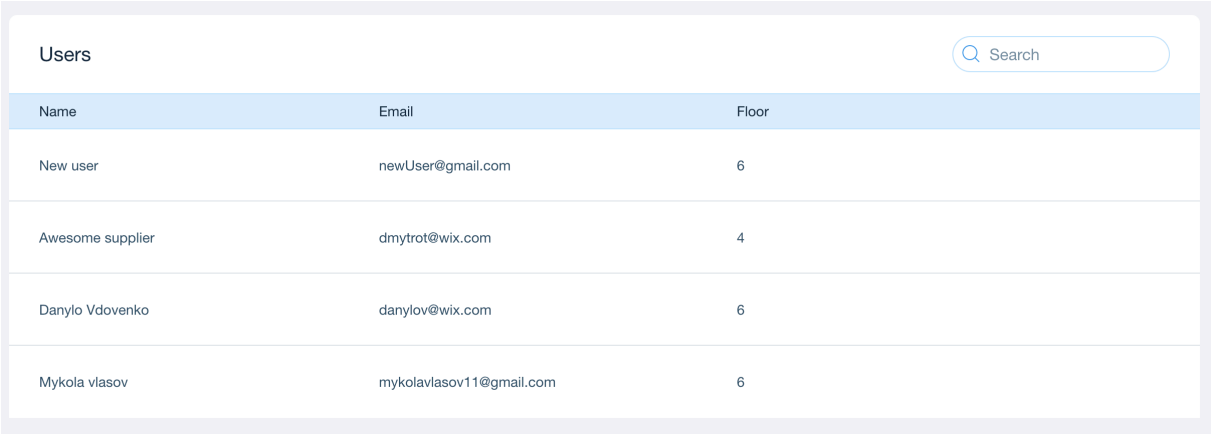


Рис. 8. Таблиця користувачів системи

Після натискання на кнопку «Suppliers» відображається відповідна форма для додавання постачальника (рис. 9) та відповідна таблиця для перегляду вже існуючих постачальників (рис. 10).

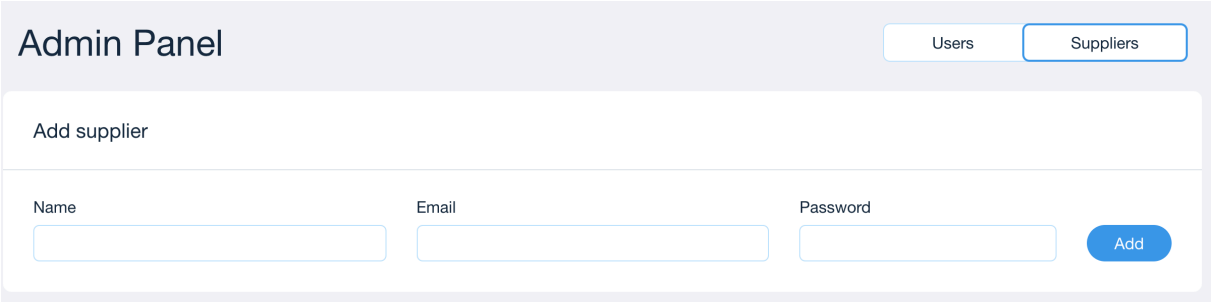


Рис. 9. Форма додавання нового користувача

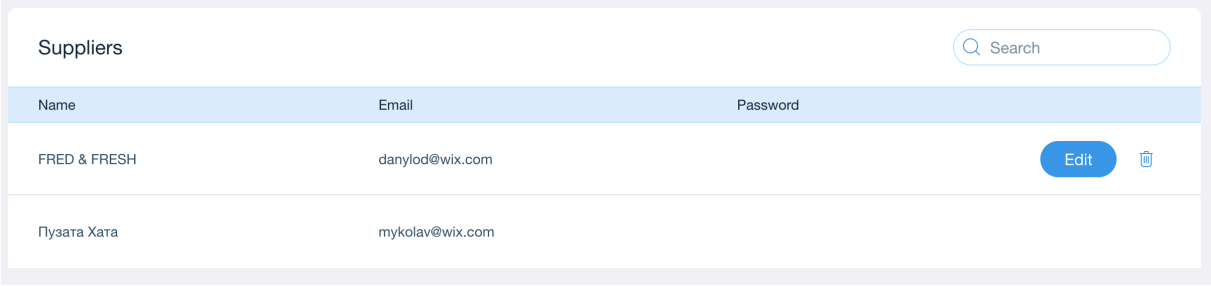


Рис. 10. Таблиця постачальників системи

6. Процедура редагування списку кулінарних виробів

Після успішної авторизації, користувач який був ідентифікований як постачальник потрапляє на відповідну особисту сторінку (рис. 11). Сторінка постачальника складається зі списку його кулнірних виробів.

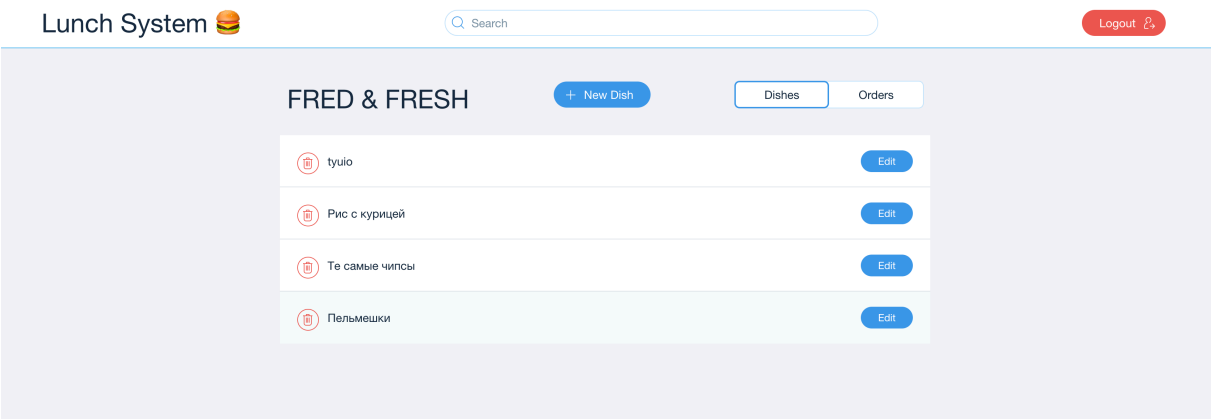


Рис. 11. Сторінка перегляду списку кулінарних виробів

Після натискання на кнопку «Edit» елемента списку, відображається форма редагування кулінарного виробу з вже заповненою інформацією про нього (рис. 12). Після закінчення редагування потрібно натиснути на кнопку «Save» для збереження змін.

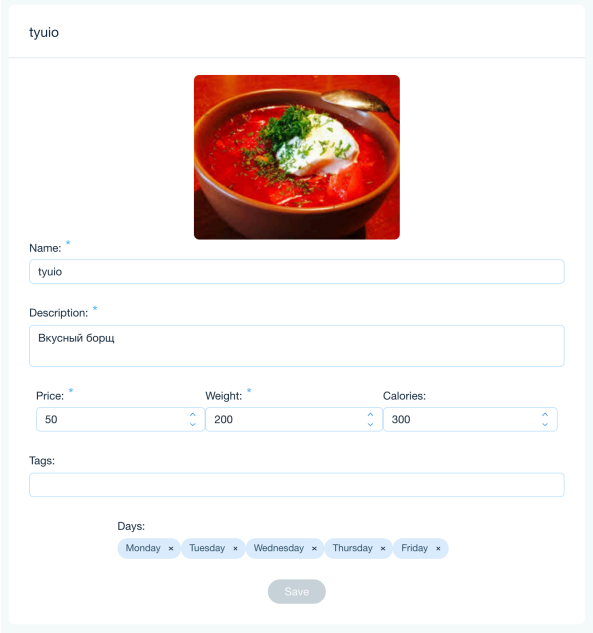


Рис. 12. Сторінка постачальника